A Geant4 model of the proton therapy beamline at the Clatterbridge Cancer Centre

Department of Physics & Astronomy,
University College London

Master's Project

27th September 2016 - 27th March 2017

Matthieu Hentz
MSci Physics

1st Supervisor: Dr Simon Jolly

2nd Supervisor: Prof Ruben Saakyan

UCL DEPARTMENT OF PHYSICS AND ASTRONOMY

Submission of coursework for Physics and Astronomy course PHASM201/PHAS3400

Please sign, date and return this form with your coursework by the **specified deadline** to the Departmental Office.

DECLARATION OF OWNERSHIP

I confirm that I have read and understood the guidelines on plagiarism, that I understand the meaning of plagiarism and that I may be penalised for submitting work that has been plagiarised.

I confirm that all work will also be submitted electronically and that this can be checked using the JISC detection service, Turnitin®.

I understand that the work cannot be assessed unless both hard copy and electronic versions of the work are handed in.

I declare that all material presented in the accompanying work is entirely my own work except where explicitly and individually indicated and that all sources used in its preparation and all quotations are clearly cited.

Should this statement prove to be untrue, I recognise the right of the Board of Examiners to recommend what action should be taken in line with UCL's regulations.

Signed [illegible]

Print Name Matthieu Hentz

Dated Monday 27th March 2017

Title	Date Received	Examiner	Examiner's Signature	MARK

Abstract

The proton beam therapy (PBT) group at UCL is tasked with developing a compact detector with a high energy resolution of 1% or better and fast response time to speed up quality assurance procedures for PBT. The prototype is regularly tested using the cyclotron located at the Clatterbridge Cancer Centre in Liverpool, the only PBT facility in the UK. The aim of this project was to develop a full Geant4 model of the 60.0 MeV beamline and to characterise the model to determine if it is behaving as expected. A total of six quantities were considered; the transverse profile and its projection onto the x-axis, the beam emittance, the kinetic energy spectrum and the lateral and longitudinal dose depositions in water. The model was found to behave as expected qualitatively, however the quantitative analysis is not yet satisfactory in that the final energy of the beam is slightly too high at 60.14 MeV, and the lateral dose deposition is not uniform enough. This could simply be a matter of verifying the input parameters taken from an old simulation, which have thus far not been validated. Additionally, the model may not account sufficiently for interactions between the protons and the thin layers of material in the dose monitors.

Contents

Abstract **3**

1 Introduction **7**

1.1 Radiotherapy . . . 7
1.2 Proton therapy . . . 9
1.3 Quality assurance . . . 10
1.4 Aim of the project . . . 11

2 Geant4 **12**

2.1 How does it work? . . . 12
2.2 Setting parameters . . . 15
2.3 The physics . . . 16
2.4 The particle source . . . 16
2.5 Defining the geometry & materials . . . 17
2.5.1 Boxes & tubes . . . 18
2.5.2 Subtraction volumes . . . 18
2.6 Producing output: scoring & tracking . . . 18
2.6.1 Scoring . . . 18
2.6.2 Tracking . . . 18
2.7 Visualisation . . . 19

3 The model **21**

3.1 The Clatterbridge beamline . . . 21
3.2 The physics . . . 21
3.3 The particle source . . . 22
3.4 Materials . . . 23
3.5 The geometry . . . 23
3.5.1 Beamline . . . 24

3.5.2 First tube 25
3.5.3 First box & second tube 26
3.5.4 Second box: dosimetry 27
3.5.5 Dose monitors 28
3.5.6 Cross-wires 28
3.5.7 Detector 28
3.6 What a simulation normally looks like 30
3.7 Scoring & tracking 30
3.7.1 Scoring 30
3.7.2 Tracking 31

4 Beam characterisation 34
4.1 Measured quantities 34
4.1.1 Beam profile and its projection onto the x-axis 34
4.1.2 Emittance 34
4.1.3 Energy spectrum 35
4.1.4 Energy deposition detector 35
4.2 Results 35
4.3 Discussion 42

5 Conclusion 44
5.1 Improvements 44
5.2 Next steps 45

Bibliography 46

6 Appendices 50
6.1 Running simultaneous simulations 50
6.2 Macros 51
6.2.1 `proton.mac` 51
6.2.2 `proton.mac_template` 52
6.2.3 `visualisation.mac` 53
6.2.4 `score_init.mac` 54
6.2.5 `score_dump.mac` 55
6.3 Bash scripts 56
6.3.1 `submit.sh` 56

6.4 Geant4 class files 57
6.4.1 DetectorConstruction.cc 57
6.4.2 SteppingAction.cc 65
6.4.3 SteppingAction.hh 67

Chapter 1

Introduction

In 2014, about 29% of all deaths in the UK were caused by cancer [1, 2]. Hence, a large proportion of medical research is dedicated to its treatment and potential cure. The main aim of cancer treatment is the removal or destruction of cancerous cells. So far, two major approaches have been taken in tackling the disease. The more gentle one consists of managing cell behaviour or inhibiting cell growth using hormone or biological therapy, while the other is to either entirely remove or aggressively attack affected cells in order to kill them. Several factors are considered in determining the type of treatment to be used, such as ease of application and how aggressive the cancer is. Usually, different combinations of the latter three treatments are chosen, depending on the specific type of cancer at hand. Nearly half of all cancer patients undergo radiotherapy [3]. It can be administered by itself or in conjunction with chemotherapy to increase the efficacy of the treatment. It is also used to keep incurable cancers in check.

1.1 Radiotherapy

Very soon after the discovery of X-rays by Wilhelm C. Röntgen in 1895 [4], their potential for application in the medical field was recognised [5]. Famously, when investigating the "new kind of rays" he had just discovered, Röntgen used a vacuum tube to produce an X-ray image of his wife's hand showing nothing but her skeleton and ring [6]. Although first attempts at curing cancer using radiation closely followed the discovery of X-rays and radioactivity (1896), it took another 27 years before Henri Coutard showed that certain types of cancer could successfully be treated using a form of radiation therapy called brachytherapy—the insertion of so called radioactive *applicators* into, or around, cancerous tissue—at the International Congress of Oncology in Paris in 1922 [7]. Biological effects of X-rays on human tissue had been widely observed in the form of burns and radiation poisoning as a result of unprotected handling of

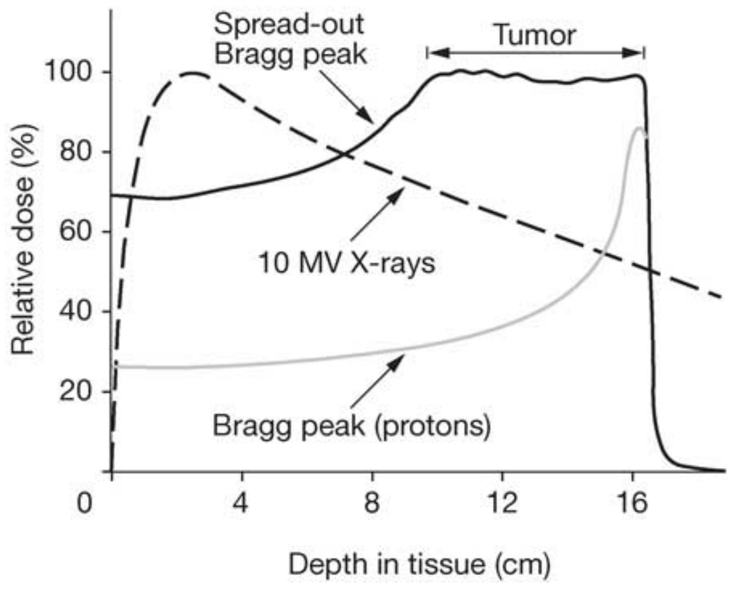

Figure 1.1: Depth dose distribution in tissue of a single Bragg peak (grey), a spread-out Bragg peak (black), which consists of multiple Bragg peaks of different energy added together, and a 10 MV X-ray beam (dashed) [8].

radiative equipment. Only now was it known how this highly energetic radiation affected human cells: the high energy photons contained in the X-ray beam deposit a given dose (energy per unit mass) in the tissue causing *ionising events*. When radiation carrying enough energy travels through matter, occasional collisions cause electrons to be freed from atoms or molecules in its path. These events can cause DNA damage which is thought to kill any cell receiving a sufficiently high dose [8]. These advances in the understanding of radiation paved the way for RP Paterson to develop X-ray radiotherapy as a feasible, controllable treatment for cancer, the results of which he reported in 1937 [7].

The biggest challenge in radiation therapy consists in destroying cancerous cells while maintaining the damage inflicted on normal, healthy, tissue to a minimum. The major inherent physical limitation in conventional X-ray radiotherapy lies in the profile of the depth dose distribution curve of an X-ray beam in human tissue. As shown in Fig. 1, peak dose deposition occurs at around 3 cm depth, while in this example the tumour is actually located at a depth ranging from about 9 cm to 16 cm. This obviously means that a substantial amount of the dose is actually deposited in normal tissue and due to its carcinogenic character the risk of causing radiation-induced secondary malignancies increases. Technological and procedural advances in recent years have allowed radiotherapy to be employed much more precisely than ever before, naturally reducing the dose imparted to normal tissue surrounding the tumour. An example of a modern, highly sophisticated technique is intensity-modulated radiation therapy (IMRT) which combines the use of 3D CT/MRI scans and a multi-leaf collimator [8, 9, 10] to create conformal dose distributions [8]. Such approaches aim to reduce the amount of dose deposited in normal tissue as much as possible by carefully planning irradiation of the affected areas.

1.2 Proton therapy

The use of other kinds of particles in radiation therapy had not been a feasible prospect for many years due to the short penetration depth in tissue of protons, deuterons (deuterium nuclei) and alpha particles at the then attainable energies [11]. The construction of larger accelerators capable of reaching energies ranging from 125 to 400 MeV started in the 1940s, prompting Robert Wilson to propose the use of protons in radiotherapy [11]—a procedure which has come to be known as *proton beam therapy* (PBT). It is straightforward to understand why he put forward the idea when considering the depth dose deposition profile of the proton shown in Fig. 1.1. Contrary to photons, protons do not mostly ionise tissue close to the surface. Instead, the dose deposited by the protons slowly increases as they slow down until maximum penetration depth is reached, at which point the residual energy is lost over a very short distance. The distinctive shape of the curve around 16 cm, where energy is rapidly lost, is known as the *Bragg peak* [8, 11]. If several proton beams of different, carefully selected, energies are superposed, a profile matching the curve referred to as the "spread-out Bragg peak" in Fig. 1 is achieved. As is indicated, the width of this spread-out peak corresponds to the width of the tumour inside the tissue. This means that most of the ionising events take place inside the tumour, greatly reducing the likelihood of affecting normal tissue surrounding the cancerous cells. Furthermore, it has been put forward that PBT requires smaller doses due to the biological effects of protons being 10% higher than those of X-ray photons [8]. This is particularly important in patients that are more susceptible to long-term side effects due to radiation damage. Since irradiation of normal cells inhibits their development, radiotherapy is more detrimental to children. Unfortunately, tumours of the central nervous system are the most commonly occurring paediatric tumour [8]. As previously discussed, radiation therapy is especially prevalent in cases where surgical access to the affected region is more difficult and dangerous, and is therefore a frequently used treatment for children with such brain tumours. It has been suggested that it is associated with long term developmental issues worsening with increasing radiation dose, younger age at time of treatment, and increased volume of brain tissue that is treated [8, 12]. The intrinsically more precise targetting of PBT would greatly reduce such long-term side effects and childhood development would remain unaffected. However, because many proton accelerators are currently not hospital-based, proton therapy is not widely available to the patients who need it most. The Clatterbridge Cancer Centre is the only centre in the UK that can, at present, provide proton therapy. However, the facility produces low-energy proton beams which can only be used for eye proton therapy since these protons have shorter penetration depths in tissue (31 mm at 60 MeV) [13]. To provide proton beam therapy where it is needed most, in 2013 the UK government approved £250 million worth of funding for high-energy proton beam centres to be built at University College London Hospital (UCLH) and The Christie in Manchester [14, 15]. The UCL high-energy physics group is undertaking R&D on proton calorimetry to be used at UCLH.

1.3 Quality assurance

In a proton beam therapy facility, protons are accelerated to the required energies in a particle accelerator and delivered through a beamline mounted to a gantry. These gantries are over three storeys tall, weigh more than a hundred tonnes and must rotate around the patient to precisely deliver the beam from any angle [16]. The complexity of the equipment requires stringent quality assurance (QA) procedures to be carried out every day before treatment begins in order to guarantee its safe operation. Most of the time carrying out these procedures is spent checking that the protons penetrate the correct depth at several energies. Energy QA measurements take a significant amount of time to set up and adjust for different energies leading to the full procedure taking up to an hour. Ideally, QA should only take a few minutes and should be easy to carry out such that more time can be spent on treating patients. In order to speed the verification of proton energy measurements, a faster and more accurate detector is required. UCL is currently tasked with designing a proton calorimeter that has a higher energy resolution, is more stable and has a faster response time than any other existing device.

The SuperNEMO high energy physics experiment requires high resolution calorimeters to investigate rare radioactive decays that, when observed, can be used to restrict bounds on the neutrino mass [17]. The design of a calorimeter unit developed for SuperNEMO consisting of a scintillator block with a hemispherical cut-out to accommodate the photomultiplier tube (PMT) has been modified to record the energy of a proton beam. The protons are absorbed by a plastic scintillator which converts their kinetic energy into light detected using a PMT. From the PMTs signal, the energy can be measured and the range of the particles calculated. The scintillator's inherently high light output and fast response time make it the ideal candidate to reduce the time taken to carry out the energy QA procedure. The whole detector assembly consists of the following: a 3 cm x 3 cm x 5 cm cuboid Envinet standard scintillator coupled to a 2-inch Hamamatsu R13089-100 11 PMT with a negative high voltage (HV) active divider base using BC-630 Saint Gobain silicone optical gel with a refractive index of 1.465. The scintillator is

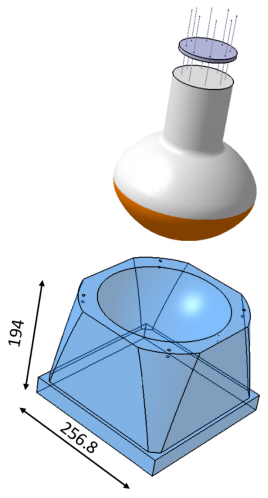

Figure 1.2: **The SuperNEMO calorimeter unit** consisting of the scintillator (blue block) and the photomultiplier tube (PMT) [17]. The measurements are given in mm.

wrapped in reflective Teflon and aluminised Mylar film in order to retain as many of the photons created by proton impacts, hence greatly improving the detector's efficiency. Ultimately, energy resolutions better than 1% should be achieved using this device, which will allow the uncertainty in the range of the proton beam to be kept at a minimum ($\sim$ 3 mm at 200 MeV [18]). Ideally, the final calorimeter module will be small enough to mount directly to a proton beam therapy treatment nozzle. With power and signal cables being the only obstruction, the whole module could then rotate with the gantry, eliminating the need to modify the setup of the QA system for different gantry angles, greatly reducing the time taken to carry out the quality assurance procedures.

1.4 Aim of the project

The National Centre for Eye Proton Therapy based at the Clatterbridge Cancer Centre in Liverpool has been the only proton therapy facility in the UK for nearly three decades. The detector prototypes being developed for QA at the new proton beam therapy centres in London and Manchester are tested there around three to four times a year. Besides the fact that the cyclotron is quite old—deteriorating its performance—the beamline has some specialised features as it was designed for treatment of ocular melanomas. The aim of this project is to develop and characterise a full model of the beamline as it would be a useful tool to help discriminate between the beam characteristics and the detector response, for example in cases where the detector may behave unexpectedly.

Chapter 2

Geant4

Written in C++, a language equipped with object-oriented programming capabilities, Geant4 is a highly versatile toolkit that uses Monte-Carlo methods to simulate the trajectories of individual particles travelling through a pre-defined geometry. It is used in many different branches of physics including high energy physics (HEP), medical physics and astrophysics. Some of its common uses are simulations of entire detector assemblies to study detector response—it is the engine of choice for ATLAS [19]—and simulations of beamlines to study beam characteristics. This section introduces Geant4 and is an attempt at describing its inner workings. Some of what is described here can be found interspersed in the Geant4 user documentation [20] to varying degrees of clarity. Version **10.2.1** was used in this project.

2.1 How does it work?

When a simulation is executed, a *run manager* represented by a `G4RunManager` class object is instantiated. It manages all the procedures in a *run*, itself represented by a `G4Run` object. In principle a simulation can contain any number of consecutive runs. However, in this project each simulation corresponds to a single run, hence the terms "simulation" and "run" are used interchangeably. The run manager first constructs the *geometry* of the system that can include either a detector, a beamline, or both, defined in the `G4VUserDetectorConstruction` class. This is followed by the initialisation of `G4VUserPhysicsList`, a class that accommodates *particles* and *physics processes* represented by `G4ParticleDefinition` (rest masses, lifetimes, charges, process cut-offs) and `G4VProcess` (tables of cross sections, energy losses) respectively. Depending on the interaction it describes, a process possesses one or more of the following actions [21]:

– *at rest* where an `AtRestDoIt()` function is called when particles are at rest (e.g. decay at rest);

- *along step* where an `AlongStepDoIt()` function implements behaviour that is "continuously" occurring along a step such as energy loss or secondary particle production;
- *post step* where a `PostStepDoIt()` function is invoked at the end of the step (secondary particle production by a decay or interaction, for example).

Multiple *continuous* processes are invoked at the same time while only one *discrete* process can handle the end of the step. This can be easily understood in terms of the following example; it should be clear that a proton cannot simultaneously undergo elastic and inelastic scattering, both discrete processes. Finally, *primary particles*—or primaries—are generated according to the parameters set in `G4VUserPrimaryGeneratorAction`, including position of the source, direction of travel of primary particles, spatial and/or angular distribution amongst others. Only at this point is `G4Run` instantiated. Each run consists of a chosen number of *events* represented by `G4Event` objects and controlled by the *event manager* class `G4EventManager`. An event has associated with it instances of `G4PrimaryParticle` containing the description of each primary particle (rest mass, lifetime, charge, momentum) and primary vertices `G4PrimaryVertex` with the time and position information of the different primary particles. Similarly as with a run, any number of primary particles can constitute an event but in this project each event corresponds to a single primary proton and hence contains only one primary vertex. Events are processed sequentially so the protons are transported one-by-one through the geometry from the position of the *source*, i.e. the position of the primaries generator, until they are eventually "killed" as they have exited the *world volume* (see Section 3.5 below), come to rest via the loss of energy through different scattering events, or been lost through decay or inelastic scattering. During the simulation, the properties of the particle being transported can be accessed through a *track*—an instance of `G4Track` which stores the current particle's definition, the ID of the parent particle (useful when dealing with secondary particles), and the `G4VPhysicalVolume` the track is currently located in (see Section 2.1.11) amongst the more useful variables. Most importantly though, the track carries the `G4Step` class object of the current *step*—the smallest unit of simulation in GEANT4. The algorithm to compute a step is given below [22].

Stepping algorithm

1. If the particle comes to a halt (zero kinetic energy), each active process with an `atRest()` function proposes a step length based on the interaction it describes. The process proposing the shortest step length will be invoked.

2. If the particle is not at rest, each active process proposes a step length and the smallest of these step lengths is taken.
3. The "safety" distance to the next volume boundary is calculated. If the step length from the previously invoked process is shorter than this distance, it is selected for the next step.
4. If the step length from the process is longer than the safety, the distance to the next boundary is re-calculated.
5. The smaller of the process step length and the distance to the nearest boundary is taken for the next step length.
6. All active continuous processes are invoked. The particle's kinetic energy is only updated after all invoked processes have completed. The change in kinetic energy is the sum of all contributions from these processes.
7. The kinetic energy, position and time of the current particle are updated before the appropriate discrete process is invoked. Simultaneously, secondary particles created by the continuous processes are stored.
8. The kinetic energy is checked to determine if the particle has been terminated by a continuous process. If the kinetic energy has been reduced to zero, `atRest`-processes will be applied at the next step if applicable.
9. The appropriate discrete process is invoked. Afterwards the energy, position and time of the particle are updated and again any secondary particles created are stored.
10. The track is checked for whether it has been terminated by the discrete process.
11. The "safety" distance to the nearest boundary is updated.
12. If the step was limited by the nearest boundary, the particle is pushed into the volume that boundary belongs to.
13. Hit information is processed if scorers are used (see Section 2.6.1).
14. The user-defined `G4UserSteppingAction` is called to perform tasks specific to each simulation.
15. Data are saved to a `G4Trajectory` object which can later-on be used to visualise the trajectories of all the particles.
16. The mean free paths of the discrete processes are updated.

17. If the parent particle is still alive, the maximum interaction length of the discrete process which has occurred is reset.

18. The step is completed.

The `G4Step` object contains the endpoints of the step, `PreStepPoint` and `PostStepPoint`, which in turn hold useful information such as the volume they are in, the material of that volume, the current position, and various kinetic quantities. This is particularly useful since, as shown in step 14 of the stepping algorithm above, each step can be accessed through the “user hook” `G4UserSteppingAction` if required. This class can be used to, for example, calculate the energy deposited in each layer of a segmented detector, or as described later-on (see Section 2.6.2), record the position of each particle along with its energy and angular orientation. Both, `G4Track` and `G4Step`, are said to be “transient”—they only contain information for the current step and are updated for each new step, hence they are not available at the end of an event. Data can be extracted in a multitude of ways; in this report we will discuss writing data to an output file using scorers and particle tracking in `G4UserSteppingAction`. A third alternative involving so-called *sensitive detectors* is discussed in Section 5.1.

2.2 Setting parameters

In the above section it was alluded to several times that parameters can, or in most cases must, be set by the user. There are three ways this can be done:

– Parameters can be set directly in the desired class, e.g. one could set all the desired properties of the primary particles (e.g. initial energy distribution, transverse spatial and angular distribution) directly in the `G4VUserPrimaryGeneratorAction`.

– Geant4 allows for two modes of operation: an *interactive* mode and a *batch* mode. In interactive mode, the user is required to set parameters using commands entered into a graphical user interface (GUI). Many classes come with pre-defined commands such as `/run/beamOn` which is used to set the number of events to be simulated, or `/gps/particle` which sets the type of primary particle, just to name a couple. Additional commands can be defined for user-defined classes in specially dedicated *messenger* classes. An example of such a command is `/steppingAction/tracking` which was defined in `G4VUserSteppingActionMessenger` to turn tracking on and off with the additional option to set a specific position for tracking.

– In batch mode, the commands are specified in a *macro* file `proton.mac` (see Appendix 6.2.1) which is then read at the start of the simulation and sets all the desired parameters.

In practice, some parameters are set directly in classes while most are set in the macro. All simulations performed for this project used the batch mode.

2.3 The physics

As mentioned in Section 2.1, the particles and physics used in the simulation are defined in `G4VUserPhysicsList`. Geant4 comes with a set of physics lists which can be selected in the macro with the user-defined command `/protonB/phys/addPhysics` provided the `AddPhysics-List()` function in the physics list class was adapted accordingly (see Section 3.2). Customised lists can be created by combining the built-in lists in various combinations. Different combinations may be required to optimise the accuracy of the simulation in different energy regimes. The physics lists are regularly checked and updated with each release, ensuring the applicability of Geant4.

2.4 The particle source

The source of primary particles is defined in the `G4VUserPrimaryGeneratorAction`. In this project a `G4GeneralParticleSource` was used because it allows control of the following characteristics:

– The *spatial distribution* of the source can be uniformly distributed over a range of shapes defined in the macro using the `/gps/pos/shape` command which is only applicable if the *type* of the beam is set to “point”, “plane”, “surface” or “volume” using `/gps/pos/type`. To use a non-uniform distribution, the beam must be chosen to be of type “beam”. Then, the particles are distributed according to a Gaussian with standard deviations in x and y specified by the commands `/gps/pos/sigma_x` and `/gps/pos/sigma_y`.

– The *angular distribution* can similarly be defined. The types available are “iso” for spherically isotropic emission, “cos” for a cosine emission law, “planar” for planes with fixed wave momentum vector, and “beam1d” or “beam2d” for a Gaussian distribution. “beam1d” allows the user to set a standard deviation symmetric in x and y while using “beam2d” lets the user define separate standard deviations for the two transverse axes. Analogously to the spatial distribution, the type is chosen using `/gps/ang/type` and the standard deviations are set with `/gps/ang/sigma_x` and `/gps/ang/sigma_y`.

– The *energy distribution* can be chosen to be; linear, exponential, a power-law, Gaussian, thermal Bremsstrahlung, a black-body spectrum at temperature T, or a diffuse cosmic gamma-ray spectrum. Again, the distribution to be used is chosen using `/gps/ene/type` and the standard deviation is set with `/gps/ene/sigma` when applicable. The desired mean energy is then set using `/gps/ene/mono`.

Additionally, the type of particle used is chosen with `/gps/particle`, the number of particles per event is defined using `/gps/number`, and the position of the source is set using the command `/gps/position`.

A peculiarity worth noting is that the code for the particle sources does not seem to be consistent with the conventions otherwise used in the GEANT4 toolkit. When using an angular distribution, the default direction of travel is along the x-axis, while the rest of the package assumes motion in the z-direction as is conventionally the case in HEP. This can be remedied by aligning the axes used by the angular distribution with those used in the rest of the simulation by rotating the former using the `/gps/ang/rot1` command where the parameters "-1 0 0" should achieve the desired behaviour. More commands are available in `/gps` but are not detailed here as they were not used in the simulations—a full list can be found in [23].

2.5 Defining the geometry & materials

The geometry and the materials to be used in the simulation are defined in `G4VUserDetector-Construction`. Despite its name, this class is not limited to the definition of a detector. In this project for instance, it was used to model a beamline.

All materials to be used in the model must first be declared in the `DefineMaterials()` function. Before defining a new material, the required chemical elements need to be defined as `G4Element` objects given their atomic number and molar mass (see line 70 in Appendix 6.4.1). A material represented by `G4Material` can then be built by adding each element individually and specifying the material's density and number of constituent elements. A simpler, but not always applicable, method is to build a material from the GEANT4 material database [24] by invoking an instance of the `G4NistManager` (line 65 in Appendix 6.4.1).

Each individual component must be defined in `ConstructVolumes()` and normally consists of three parts: a *solid* `G4VSolid` which contains the shape and dimensions of the component, a *logical volume* `G4LogicalVolume` which assigns a material to the solid, and a *physical volume* `G4VPhysicalVolume` that places the logical volume inside the required volume. Every volume in the model is made up of either a `G4Box` box, a `G4Tubs` tube, or a combination of the two.

2.5.1 Boxes & tubes

A `G4Box` is specified using the "half-width" in all three dimensions. In other words, if the user wishes to give a box a length of, say, 10 mm in x, then the input parameter should be given as 5 mm. On the other hand, `G4Tubs` is a simple cylinder specified using the inner and outer radii, as well as its height. Again, the height must be given as a "half-length" such that a 5 mm input would result in a cylinder 10 mm tall.

2.5.2 Subtraction volumes

It might sometimes be useful to be able to subtract solids from one another. For example, to create a hollow tube with walls of an appreciable thickness one could subtract a solid tube from another, slightly larger, solid tube resulting in a hollow tube with wall thickness equal to the difference in radii of the two initial tubes. The "void" would then be filled with the same material the mother volume is filled with.

2.6 Producing output: scoring & tracking

The two methods used to extract useful information from the simulation are set out below. Note that there may well be other ways of outputting data but the choice of the two methods will be elucidated here.

2.6.1 Scoring

Primitive scorers can be used to record quantities such as the energy or dose deposited in a volume, the number of secondaries created in it, or even the particle flux through it [25]. A scorer is necessarily a box as it is created using the `/score/create/boxMesh` command in the `proton.mac` macro. The size of the scorer and number of bins in the mesh in each dimension is then specified with `/score/mesh/boxSize` and `/score/mesh/nBin`. It can be moved to the desired position and orientation using `/score/mesh/translate` and `/score/mesh/rotate`. The quantity to be scored should be set in `/score/quantity` and it is finally dumped to a file when the `/score/dumpQuantityToFile` command is called in the macro.

2.6.2 Tracking

Tracking refers to the process of recording particles in the `G4VUserSteppingAction`. It is required as the primitive scorers available do not allow for the precise position, direction of momentum, or even the kinetic energy of particles to be recorded as these quantities are not available as "scorable" quantities. This limitation may be overcome by defining custom scorers, however this was only recognised as a possibility in the later stages of the project when tracking had already been successfully implemented so changing this was not deemed necessary.

As shown in the stepping algorithm in Section 2.1, the `G4VUserSteppingAction` is called towards the end of each step in the simulation. In it, the current `G4Step` can be accessed and various quantities from its endpoints, `PreStepPoint` and `PostStepPoint`, can be written to a file using the standard C++ output stream. This proved to be especially useful when recording particle properties at set intervals in z as this could be easily implemented in a loop (line 85 in Appendix 6.4.2). A particle was deemed to have crossed a boundary if the z-coordinates of `PreStepPoint` and `PostStepPoint` were in front and behind the boundary, respectively. Only then were the desired quantities written to a file containing all the particles that crossed that specific z-position.

2.7 Visualisation

The trajectories of all particles—primaries as well as secondaries—are stored for every event. As a simulation often consists of millions of events, it is not practical to visualise such a simulation; so many tracks are produced that they become indistinguishable from one another and even hide features of the geometry used. In this project anything on the order of 100 primary particles was sufficient for visualisation.

As with many other aspects of GEANT4, there are several ways in which simulations can be visualised. Here, we will focus on the DAWN renderer (see 8.3.6 DAWN in [26]) that was used to produce the geometry representations shown in Chapter 3. Visualisation can be switched on using the `/vis/open` command in the macro. To create files viewable in DAWN, the "DAWNFILE" option should be chosen. Many parameters such as the colour of different particles or the drawing style of the visualisation can be set using various `/vis` commands (shown in Appendix 6.2.3) and, to keep input files simple, the desired commands can be written to a specially dedicated file `visualisation.mac` which is itself called in `proton.mac` using the `/control/execute` command. The main drawback of using DAWN is that the viewing parameters (polar and azimuthal angles, zoom, focal point) must be set before visualisation can begin. This means that, for instance, to adjust the viewing angle, the visualisation must be closed and reopened before new parameters can be entered into the UI and the visualisation redrawn. This naturally slows down the process and takes more getting-used-to compared to other visualisers (such as OpenGL) that have click and drag capabilities. Its strength, however, lies in the quality of the high-resolution images it renders. Shown below in Fig. 2.1 is an example of two proton tracks producing electrons via ionisation of the air they are travelling through. Each step in the simulation corresponds to a segment of the track. This figure was produced by zooming in on a couple of tracks as far as possible, therefore this is the highest level of detail attainable in DAWN.

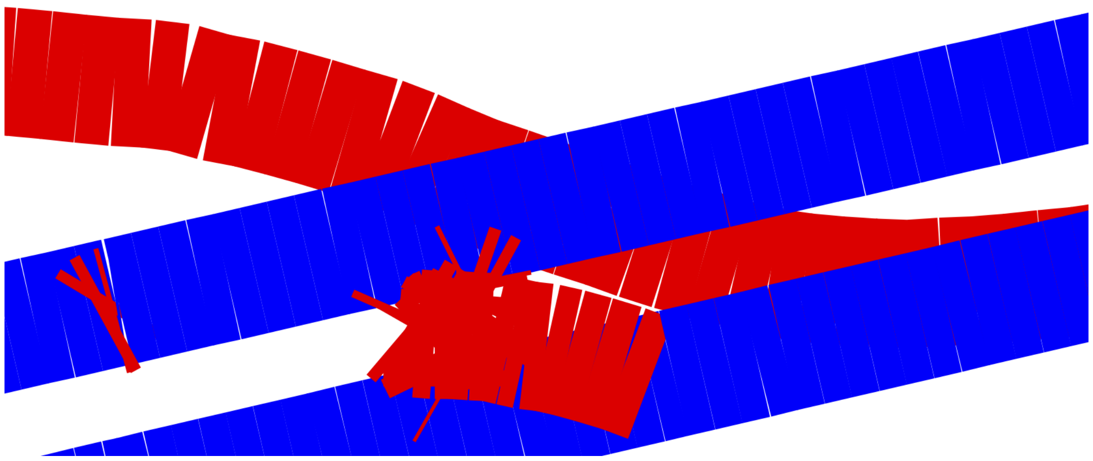

Figure 2.1: **Tracks from two different types of particle visualised in DAWN**: the blue tracks represent protons while the red tracks are electrons produced by ionisation of the air by the protons. Note that the constituent segments of the track are visible thanks to the high resolution offered by the DAWN renderer.

Chapter 3

The model

The National Centre for Eye Proton Therapy based at the Clatterbridge Cancer Centre in Liverpool has been the only proton therapy facility in the UK for nearly three decades. The detector prototypes being developed for QA at the new proton beam therapy centres in London and Manchester are tested there whenever time permits—around three to four times a year. Besides the fact that the cyclotron is quite old, deteriorating its performance, the beamline has some specialised features as it was designed for treatment of ocular melanomas. A full model of the beamline would be a useful tool to help discriminate between the beam characteristics and the detector response, for example in cases where the detector may behave unexpectedly.

3.1 The Clatterbridge beamline

The beamline at Clatterbridge produces a uniform beam that is rather wide at 34 mm, using a technique known as *dual scattering* where the pencil beam from the cyclotron is scattered using scattering foils and a brass stopper separated by a given distance. This is desirable as the beamline was designed for eye proton therapy, requiring a beam that is comparable in width to the human eye. With a beam at 60 MeV, it can penetrate water to a maximum depth of 31 mm [27].

3.2 The physics

The `QGSP_BIC_EMY` physics list was used in this simulation. It can be found in the "iort_therapy" example provided with the GEANT4 source code. The `QGSP_BIC` list is intended for the simulation of high energy physics involving protons, neutrons and nuclei but since we are interested in processes where the beam has relatively low energy, it was extended in `G4VUserPhysicsList` as follows:

```
void G4VUserPhysicsList::AddPhysicsList(const G4String& name)
{
  ...
  if (name == "QGSP_BIC_EMY")
  {
    AddPhysicsList("emstandard_opt3");
    fHadronPhys.push_back(new G4HadronPhysicsQGSP_BIC());
    fHadronPhys.push_back(new G4EmExtraPhysics());
    fHadronPhys.push_back(new G4HadronElasticPhysics());
    fHadronPhys.push_back(new G4StoppingPhysics());
    fHadronPhys.push_back(new G4IonBinaryCascadePhysics());
    fHadronPhys.push_back(new G4NeutronTrackingCut());
    G4RunManager::GetRunManager()->PhysicsHasBeenModified();
  }
  ...
}
```

This physics list also improves the performance of the simulation by limiting processes we are not directly interested in—e.g. neutron tracking.

3.3 The particle source

To produce a realistic simulation, it was important to have normally distributed initial kinetic energies, positions, and angular orientations for the primary particles. This was easily implemented using the `G4GeneralParticleSource` described in Section 2.4. The parameters shown in Table 3.1 below were taken from an old simulation and still need to be confirmed. The source was placed against the wall of the room described in the geometry Section 3.5 below using `/gps/position` and using the values "0 0 -4200 mm". As aforementioned, due to the lack of consistency with respect to the rest of the GEANT4 distribution, the axes must be rotated to

	Parameter	Value
Energy	type	Gauss
	mono	62.5 MeV
	sigma	0.082 MeV
Position	type	Beam
	sigma_x	4.0 mm
	sigma_y	4.5 mm
Angle	type	beam2d
	sigma_x	2.3 mrad
	sigma_y	1.2 mrad

Table 3.1: **Input parameters** used for the `G4GeneralParticleSource` in the simulation.

coincide with the conventional directions used elsewhere in the simulation with /gps/ang/rot1 and the values “-1 0 0”.

3.4 Materials

All the materials used in the simulation can be found in the DetectorConstructionin Appendix 6.4.1. One of the materials used is defined here as an example. Kapton is often chosen as a window material for components that must be kept under vacuum due it is high mechanical and thermal stability and relatively high resistance to radiation damage. To begin with, the required chemical elements need to be constructed given their atomic number and molar mass.

```
G4Element* H = new G4Element("Hydrogen", "H", z=1., a=1.008*g/mole);
G4Element* C = new G4Element("Carbon",   "C", z=6., a=12.010*g/mole);
G4Element* N = new G4Element("Nitrogen", "N", z=7., a=14.010*g/mole);
G4Element* O = new G4Element("Oxygen",   "O", z=8., a=16.000*g/mole);
```

Now, the desired material can be defined by its density and number of constituent chemical species and the elements are added to it while specifying their fractional mass.

```
G4Material* kapton = new G4Material("Kapton", density=1.42*g/cm3,
                                    ncomponents=4);
kapton->AddElement(H, fractionmass=0.027);
kapton->AddElement(C, fractionmass=0.691);
kapton->AddElement(N, fractionmass=0.073);
kapton->AddElement(O, fractionmass=0.209);
```

If a material from the NIST database is required, it can be obtained as it was done with steel in the following example.

```
G4Material* steel = G4NistManager::Instance()
                         ->FindOrBuildMaterial("G4_STAINLESS-STEEL");
```

3.5 The geometry

The geometry is fully defined in the DetectorConstruction.cc file which corresponds to G4VUserDetectorContruction and is shown in Appendix 6.4.1. Every volume in the geometry must be contained inside a reference volume called the *world volume*, so it should be defined first. Adhering to the procedure described in Section 2.5, the solid sWorld is the first object to be created, giving it a label and specifying the half-lengths in all three dimensions:

```
G4Box* sWorld = new G4Box("World", 9450*CLHEP::mm/2,
                          4450*CLHEP::mm/2, 9450*CLHEP::mm/2);
```

The dimensions are divided by two to make the code more legible, i.e. the world has length 9450 mm in x, so “9450/2” is easier to work with than having to remember to multiply 4725 by two every time. Next, the solid is assigned to a logical volume `lWorld` so as to associate it with the desired material.

```
G4LogicalVolume* lWorld = new G4LogicalVolume(sWorld, worldMaterial, "World");
```

The `worldMaterial` is the default material for the entire world and is chosen to be air. Therefore, any volume that is the result of a `G4SubtractionSolid` will have air-filled voids. Finally, the logical volume **must** be placed at the origin by assigning it to the physical volume `pWorld`.

```
G4VPhysicalVolume* pWorld = new G4PVPlacement(0, G4ThreeVector(), lWorld,
                                              "World", 0, false, 0);
```

The arguments in order of appearance are; (1) a rotation matrix, (2) the position vector of the centre of the volume, (3) the logical volume to be placed, (4) a label, (5) the logical volume the physical volume is being placed into, (6) a boolean not yet implemented for physical volumes in Geant4, and (7) an arbitrary index. Note that a volume containing other volumes is said to be their *mother* while the volumes contained within it are said to be its *daughters*. The mother must be set to “0” for the world volume as it is not placed in any other volume.

3.5.1 Beamline

The Clatterbridge beamline essentially consists of two aluminium tubes and boxes. Shown below in Fig. 3.1 is a top-down view of the beamline and in Fig. 3.2 a perspective view. The beam is spread out passively using *dual scattering*, where it is scattered twice through scattering foils in the first tube of the beamline. This leads to a uniform profile and flat energy distribution—the

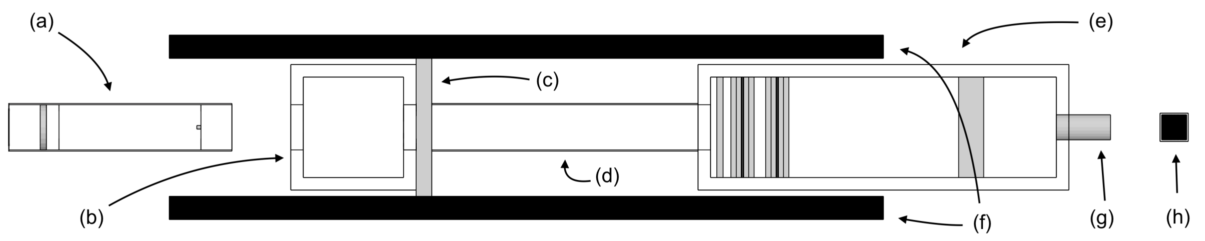

Figure 3.1: **A top-down view of the Clatterbridge beamline** visualised using DAWN. The labels correspond to the following components; (a) 1st aluminium tube, (b) 1st aluminium box, (c) iron block, (d) 2nd aluminium tube, (e) 2nd aluminium box, (f) protective shielding, (g) nozzle, (h) detector. The first box is empty as it normally accommodates a range shifter used to modulate the beam.

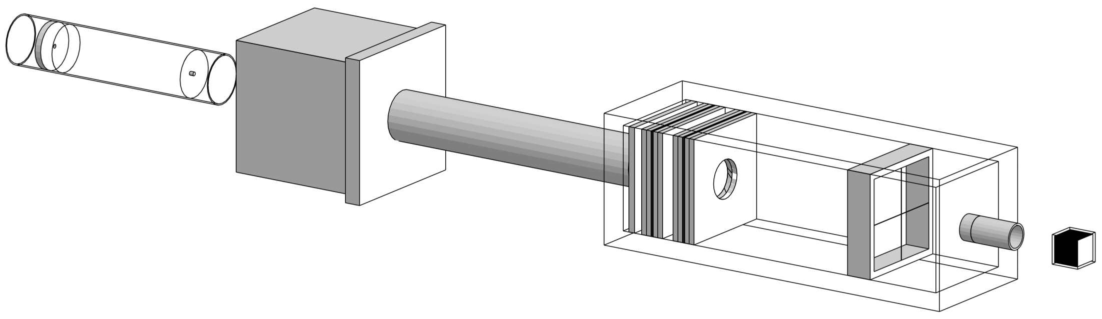

Figure 3.2: **A side-view of the beamline** without the shielding panels. The first tube and second box are visualised in *wire-frame* mode such that their inner components are visible. The grey parts are all viewed in *surface* mode.

energy deposited should be independent of lateral position. The protons then carry on through the rest of the beamline until the particles remaining after collimation terminate in the detector. Slabs of borated plastic on either side of the beamline act as shielding and should absorb any scattered protons and secondary particles—particularly neutrons.

3.5.2 First tube

The first tube is illustrated in Fig. 3.3. The beam is collimated using a brass collimator of diameter 6 mm and then spread out with a thin 25 μm scattering foil. Tungsten is used as it has a high atomic number, making it highly scattering. The spread in the beam is not immediately noticeable since the scattering foil actually spreads out the angular distribution of the beam transverse to the direction of travel. This leads to a wider beam further down the beamline, where the particles have had time to travel away from the central axis. After the first foil, the beam travels further down the first tube until it hits the brass stopper that has a diameter of 5.71 mm and continues through the second scattering foil immediately behind it. This has a much more dramatic effect than the previous scattering foil as it cuts out just less than half of the total number of protons in the beam. It is done to avoid a high proton number density at the centre of the beam. The protons exit the first tube that is kept under a vacuum through the 50 μm-thick Kapton window. This process is illustrated in Fig. 3.4 on the next page.

Due to the requirement of the tube having to be kept under a vacuum, rather than using a subtraction solid, it is constructed by placing a "tube of vacuum" inside a solid aluminium cylinder—see lines 243 to 255 in Appendix 6.4.1.

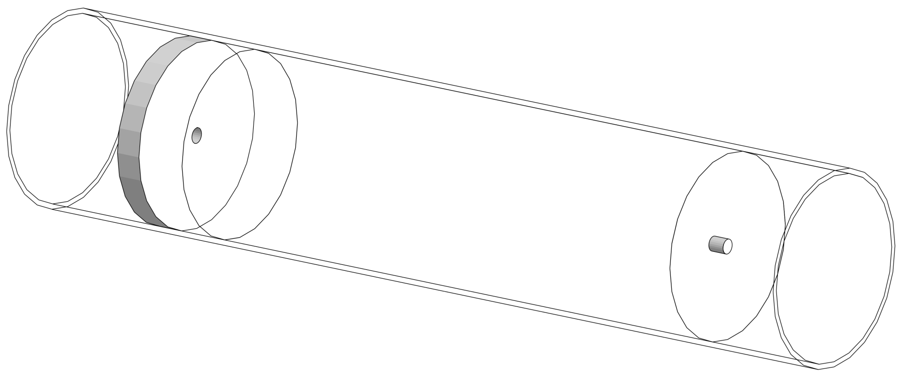

Figure 3.3: **Close-up of the first tube**. The labels correspond to the following components; (a) proton source, (b) brass collimator, (c) 1st tungsten scattering foil, (d) brass stopper, (e) 2nd tungsten scattering foil, (f) Kapton window. The protons travel from left to right.

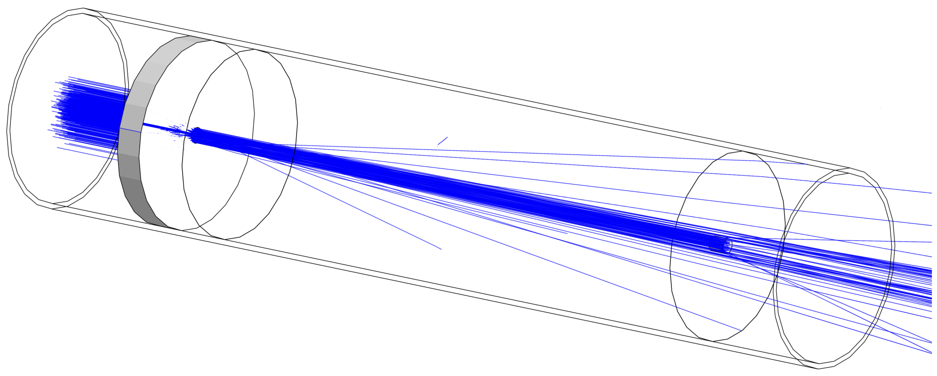

Figure 3.4: **Close-up of the first tube *in action* (500 initial protons)**. The beam is collimated and can be seen to spread out. Note that the spread is much less than this illustration might suggest since the overlapping trajectories make it difficult to get a sense of the numbers of particles involved. The colour plots in Section 4.2 should be consulted to gain an understanding of the shape of the beam at various stages in the beamline. Despite this, the effect of the beam stopper is already recognisable in this simulation with relatively few events.

3.5.3 First box & second tube

The first box remains empty in the simulation. It would normally contain a range shifter [27] which is a piece of PMMA with a given *water equivalent thickness*, or WET [28], used to lower the energy of the beam. The WET corresponds to the thickness of water required to attenuate the beam in exactly the same manner as the material it traverses. A propeller-shaped modulator wheel would also be placed in the box. Made of PMMA as well, it has blades of different thicknesses that rotate at a carefully selected rate. Each blade moves the Bragg peak back in the target volume, so as to eventually produce a spread-out Bragg peak (see Fig. 1.1). The second tube is also empty as it merely connects the two aluminium boxes.

3.5.4 Second box: dosimetry

In the second box, shown in Fig. 3.5 below, the beam is again collimated with a brass collimator of diameter 20 mm. The protons then travel through the dose monitors (close-up in Fig. 3.7) and past the tungsten cross-wires of diameter 0.4 mm before reaching the nozzle which has a diameter of 34 mm. It is expected that none of these components affect the beam significantly as that would defeat their purpose of measuring beam properties without disturbing it. The nozzle should, however, remove any particles further than 17 mm away from the axis (illustrated in Fig. 3.6), resulting in a sharp circular beam profile (see Fig. 4.8).

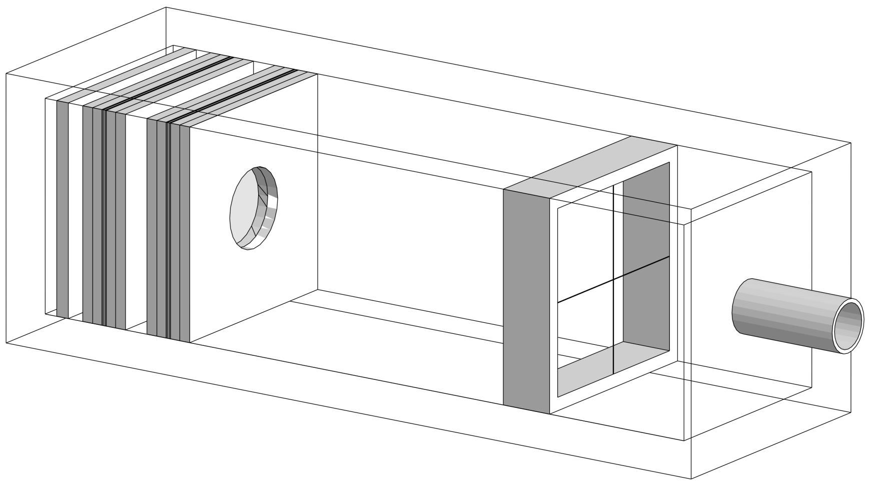

Figure 3.5: **Close-up of the second box**. The labels correspond to the following components; (a) brass collimator, (b) dose monitors, (c) cross-wires, (d) nozzle.

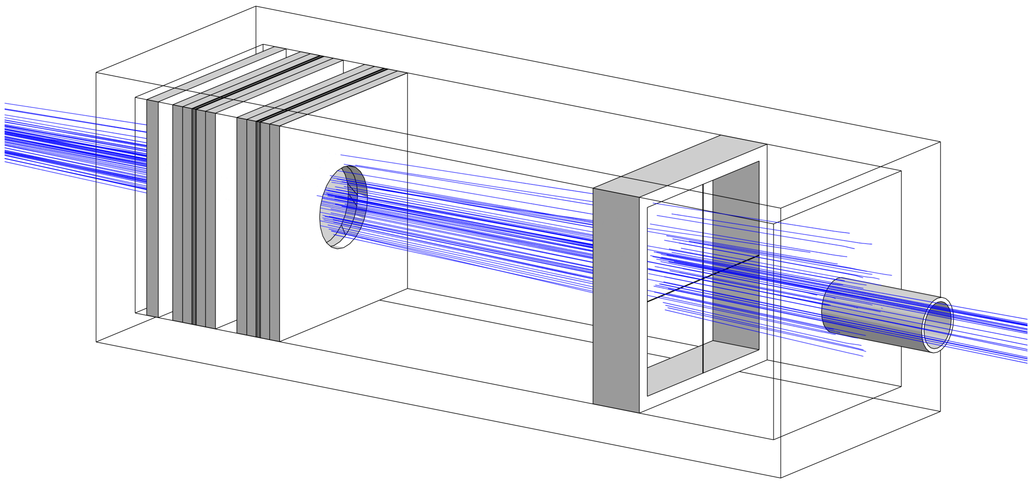

Figure 3.6: **Close-up of the second box *in action* (500 initial protons)**. The beam is now noticeably wider. As desired, the components in the box do not seem to affect the beam considerably and any proton travelling further than 17 mm from the central axis are blocked by the aluminium wall before the beam exits the box.

3.5.5 Dose monitors

The dose monitors were designed to minimise their effect on the beam allowing for *in vivo* dosimetry. They fundamentally consists of two thin layers of aluminised Mylar foil (1 μm aluminium on 5 μm Mylar) separated by a volume of air. When a potential difference is applied this behaves as an ionisation chamber. The aluminium side of the foils must be oriented towards each other such that an electric field can be established. An exploded view of a dose monitor is shown in Fig. 3.7 below—see Fig. 3.5 for an illustration of the whole component.

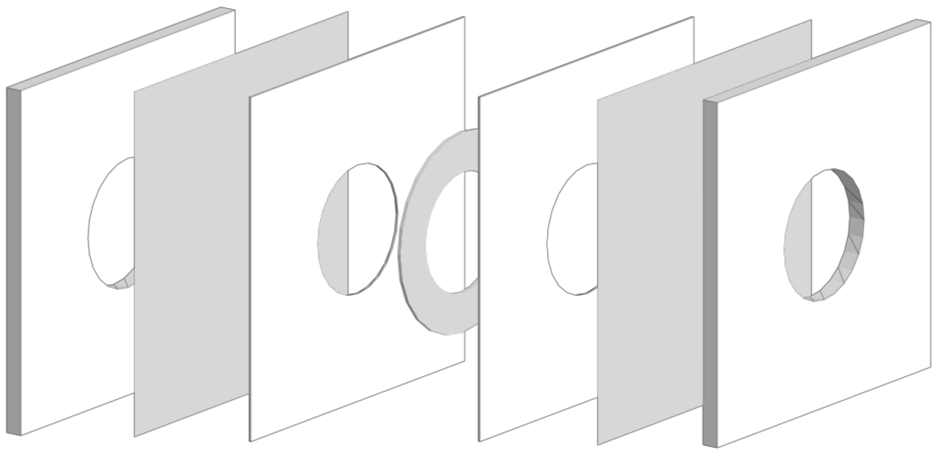

Figure 3.7: **Exploded view of a dose monitor**. From left to right the components are; a block of perspex of width 8 mm, a layer of aluminised Mylar foil with the aluminium side oriented to the right (6 μm overall), a layer of perspex of width 1 mm, a guard ring of width 1.6 mm to create a sealed volume of air between the foils, another layer of perspex of width 1 mm, a layer of aluminised Mylar foil with the aluminium side oriented to the left (again 6 μm), and finally another block of perspex of width 8 mm. All the holes have a radius of 30 mm.

3.5.6 Cross-wires

The transverse profile of the beam can be measured using tungsten cross-wires (0.4 mm diameter). The position of the protons along the x and y axes is determined from the electrical current produced when the wires are hit. The wires are mounted onto an aluminium frame as shown in Fig. 3.8 so that the point at which they cross coincides with the origin of the xy-plane of the system.

3.5.7 Detector

The detector consists of a scintillator coupled to a photomultiplier tube (PMT). The number of photons produced by a particle traversing it is proportional to the energy it deposited. If the scintillator is large enough to accommodate the depth of the Bragg peak, a measurement

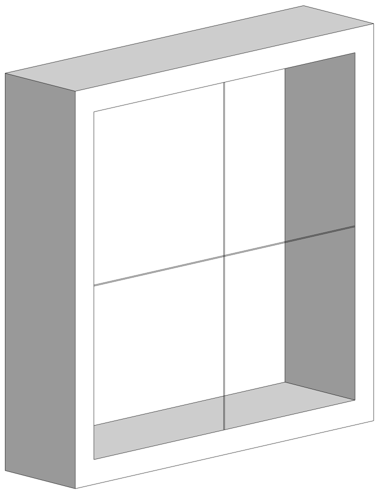

Figure 3.8: **Cross-wires fixture**. The wires are made of tungsten and have a diameter of 0.4 mm. They are used to measure the transverse profile of the beam.

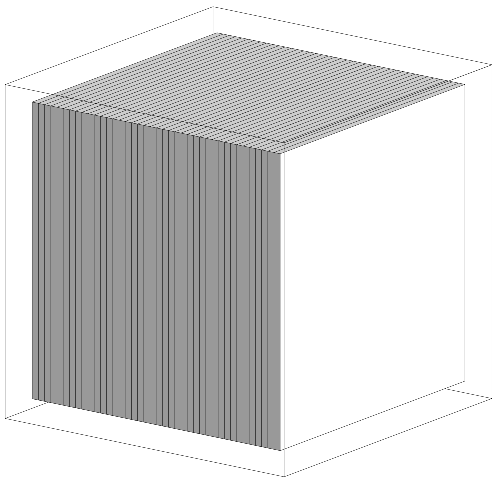

Figure 3.9: **The detector volume**. The absorber volume is the smaller cube. It is segmented such that the energy deposited in each layer can be accessed within GEANT4. Without segmentation, a scorer with a given number of bins in z could be used to determine the energy deposited in each layer but would only be available once the run has completed. The dimensions of the absorber and position of the detector volume can be defined in the `proton.mac` file.

of the photons produced can be used to infer the energy and hence the range of the incoming particle. In the model, the absorber volume is a cube of dimensions 40 mm x 40 mm x 40 mm. It is placed within an empty volume (see Fig. 3.9) whose position can be set in the `proton.mac` file. This is useful in cases where the absorber may be wrapped in a given material—e.g. Mylar or Teflon—to maximise the number of photons detected by the PMT it is coupled with, since then all the components can be moved together using only a single command.

3.6 What a simulation normally looks like

The figures containing particle tracks shown in the above section only show the protons in blue. Electrons, positrons, neutrons and gamma photons have been removed to make the figures clearer. A snapshot from a simulation where these were not hidden is shown here in Fig. 3.10.

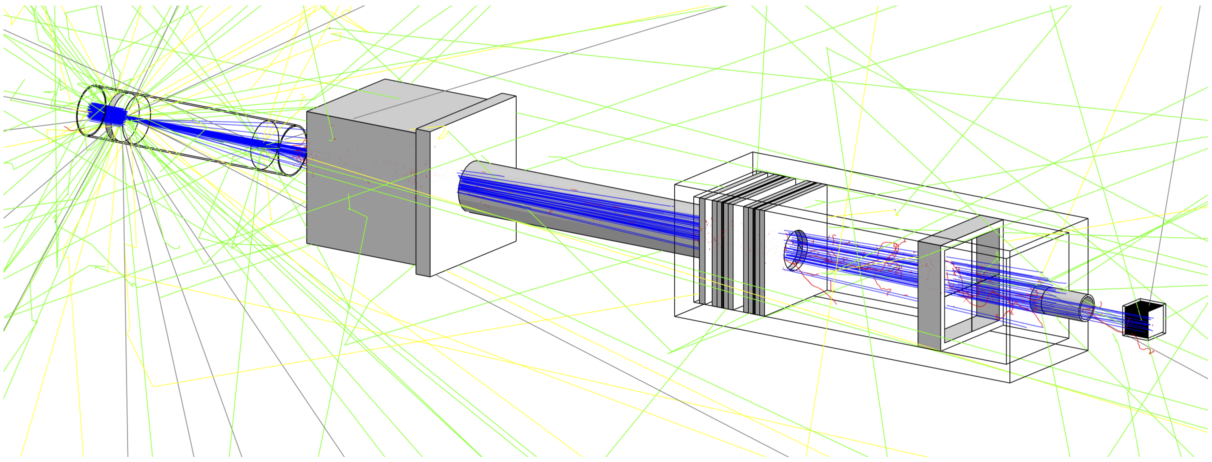

Figure 3.10: **Simulation of the full beamline** showing all the particles created. Colours are attributed to particles as follows; protons are blue, electrons are red, positrons are cyan, neutrons are yellow and gamma photons are green. Any particle without colour definition is shown as grey and is therefore unidentifiable.

3.7 Scoring & tracking

Scorers were used to record the energy deposited in given volumes and tracking was used to record the energy and the angle between the x-projection of the particles' direction of travel and the z-axis at chosen positions in the beamline.

3.7.1 Scoring

An example of a scorer used to record the longitudinal energy deposition in the detector is given below. All the scorers were defined in the macro `score_init.mac` which is shown in Appendix 6.2.4.

```
/score/create/boxMesh detEnergyLon
/score/mesh/boxSize 20. 20. 20. mm
/score/mesh/nBin 1 1 200
/score/mesh/translate/xyz 0. 0. -2340 mm
/score/quantity/energyDeposit energyDeposit
/score/quantity/doseDeposit doseDeposit
/score/close
```

The energy is typically given in MeV while the dose is given in Gy. The quantities were dumped to the files `DetEnergyDepLon.txt` and `DetDoseDepLon.txt` at the end of the run using the following commands:

```
/score/dumpQuantityToFile detEnergyLon energyDeposit DetEnergyDepLon.txt
/score/dumpQuantityToFile detEnergyLon doseDeposit   DetDoseDepLon.txt
```

3.7.2 Tracking

As previously mentioned, tracking is implemented in `G4VUserSteppingAction`. Tracking must be enabled in the macro with the custom command `/steppingAction/tracking` which sets the value of the boolean `isTracked`. The arguments can be either “All”, “None”, or a numerical value specifying the position in z where the particles are to be recorded. If the “All” option is specified, the default positions defined in `G4VUserSteppingAction.hh` are used. Seeing as every particle is processed through the stepping action, the first step is to check if the current particle is a proton:

```
G4String particleName = step->GetTrack()->GetDefinition()->GetParticleName();
bool isProton = (particleName == "proton") ? true : false;
```

Only protons inside the beamline should be tracked, so the x and y positions of the `G4PostStep-Point` are checked to see if they still fall within the boundaries. As the largest beamline components are 200 mm wide transverse to the beam (excluding the shielding), any particle exceeding a value of 100 mm in its x or y-coordinate is deemed to have exited the beamline and is not tracked. This is implemented as follows:

```
G4StepPoint* postPoint = step->GetPostStepPoint();
// use abs() as we are interested in magnitude only
G4double xposPost = abs(postPoint->GetPosition().x());
G4double yposPost = abs(postPoint->GetPosition().y());
bool isInside = (xposPost <= 100 || yposPost <= 100) ? true : false;
```

Tracking only occurs when the booleans `isTracked`, `isProton`, and `isInside` are `true`, i.e. when the following `if`-statement is satisfied:

```
if (isTracked && isProton && isInside) {...}
```

Once all the necessary conditions for tracking to occur are satisfied, the protons must be recorded accurately. Here, the challenge lies in correctly identifying whether or not a proton’s position coincides with a given position in z. As this position in z represents a plane, it can be referred to as a *boundary*. The first approach that might come to mind turns out to miscount the number of protons crossing a given boundary but is detailed here for completeness and to support the choice for the method eventually used. Both procedures have in common that the position of the boundaries must be defined in a vector of integers in the `SteppingAction.hh` class beforehand. This is to allow two different functions in `SteppingAction.cc` access to the same vector. An admittedly slightly awkward solution, it nevertheless provides an easily manageable way of recording the desired data—especially when many boundaries are used—as it allows direct control of the output format.

Comparing distance within a tolerance

If the absolute difference between a proton's actual position and a chosen boundary were to fall within a given tolerance, the particle's position and that of the boundary could be deemed to coincide. This approach quickly breaks down as a fixed tolerance would lead to overcounting in cases where the step size is smaller than it and to undercounting in cases where the step size is larger than it. Setting the tolerance equal to the maximum step size would avoid undercounting altogether but since most steps are just short off the maximum step size, overcounting would dramatically increase. Ultimately this method does not work due to the variability in the step sizes. Another approach independent of any tolerances was hence required.

Checking if boundary was *crossed*

The two end points associated with each step can be used to check if the chosen boundary was crossed by the proton. If the z-position of the particle was in front of the boundary before the step and behind it after the step, then the particle **must** have moved across the boundary and is recorded. This, naturally, also applies to particles that sit on the boundary. As the particles' "current" position in the simulation is the one in `postStepPoint`, this is the position that is written to the file. The fact that only particles sitting on the boundary or having just crossed it are saved causes a spread in the z-values ranging from the boundary to a distance equal to the maximum step size away. The full conditional used to verify boundary crossing in the simulation is shown below:

```
if (zposPre < z_boundary && zposPost >= z_boundary) {...}
```

The z-coordinate before the step is `zposPre`, after the step it is `zposPost` and `z_boundary` is the position of the boundary.

Recorded quantities

The quantities recorded for each proton are its `parentID` which is non-zero for secondary particles, its position in x, y and z, the angle x' subtended by the z-axis and the projection of its direction of travel onto the xz-plane, and its kinetic energy. Almost all of the quantities are directly accessible in the stepping action—only the angle needs to be calculated. This is done using the directional unit vector $\hat{\mathbf{r}}$ of the particle as the angle between the projection in xz and z is given by

$$x' = \frac{r_x}{r_z} \tag{3.1}$$

in the small angle approximation $\tan x' \approx x'$. Note that this can be done using the components of any vector pointing in the direction of travel, i.e. the momentum vector could also be used.

Chapter 4

Beam characterisation

As with any other model, it is important to validate the behaviour of the simulation before it can be fully trusted. This is done by monitoring quantities characterising the beam such as its profile and emittance, the number of protons recorded at a given position, its energy spectrum and the profile of the dose deposition in the detector volume. For the purposes of characterisation of the beamline, water was used as an absorber as it is the standard normally used due to human tissue—consisting in large parts of water—having a nearly identical density (1.049 kg/dm^3 for muscle tissue [29]), allowing the assumption the beam will behave similarly in tissue.

4.1 Measured quantities

4.1.1 Beam profile and its projection onto the x-axis

The transverse profile of the beam can be plotted using the xy coordinates of the protons recorded at different positions in z. To verify the positions are normally distributed—at least initially—the profiles were also projected onto the x-axis. The total number of protons remaining at a given position (shown in all projection plots) was determined by integrating the projection in x as this was the easiest method to implement. When large numbers of protons are simulated, a simple scatter plot is not sufficient to visualise the profile in any meaningful manner thus requiring the data to be binned in order to plot density plots instead.

4.1.2 Emittance

The *emittance* of a beam can be determined from the plot of the angle measured using Eq. 3.1 against the particles' position in x. As these two quantities are correlated, an ellipse of area $\pi\varepsilon$ is obtained, where ε is defined to be the beam emittance [30]. The parameters describing the shape of the ellipse—the *Twiss parameters*—can give an insight into different properties of the beam. For example, the α parameter quantifying the x-x' correlation is negative for converging beams, and positive for diverging beams. Hence, the ellipse is expected to show clockwise shearing when measured at different points moving down the z-axis along the beam since that corresponds to an increasing correlation. This concept proves to be intuitive as one would expect particles travelling in a direction pointing away from the beamline to move further away from the central axis than a particle travelling in a direction more closely aligned with it. Similarly as with the beam profile, the data are binned to create density plots.

4.1.3 Energy spectrum

The data are binned in order to plot a histogram of the energy spectrum. A measure of the energy of the beam at a given position in z is obtained by fitting a Gaussian to the corresponding histogram, yielding values for the mean and the standard deviation of the spectrum at that point.

4.1.4 Energy deposition detector

The energy deposition in the detector is expected to show a Bragg peak at a depth of around 31 mm in water as described in Section 1.2. This is done by plotting the energy deposition with respect to depth and finding the position of the maximum.

4.2 Results

The different plots characterising the beamline are shown in this section. The initial number of protons was chosen to be 10 million in order to achieve a better definition in the plots as the colour plots tend to become too faint once the beam starts to even out towards the end of the beamline. The initial beam parameters used were given in Table 3.1. In the colour plots, a light colour indicates a high density of protons, while a darker colour indicates a lower density.

At the source (Fig. 4.1), the beam has a wide distribution both in x and y and has a small range in its angular distribution—at most 5 mrad away from the central axis. 10 million protons are counted by integrating the projection and the kinetic energy lies at 65.5 ± 0.082 MeV. The blueish halo around the central portion of the beam profile disappears once it has travelled through the collimator (Fig. 4.2), leaving a circular beam profile of diameter 6 mm which matches that of the collimator. Note that roughly 76% of the original beam is cut out at this stage. This is distinctly visible in the projection—the tails have disappeared and the curve now has sharp edges. The scale in the plot of the emittance has been changed to underline the

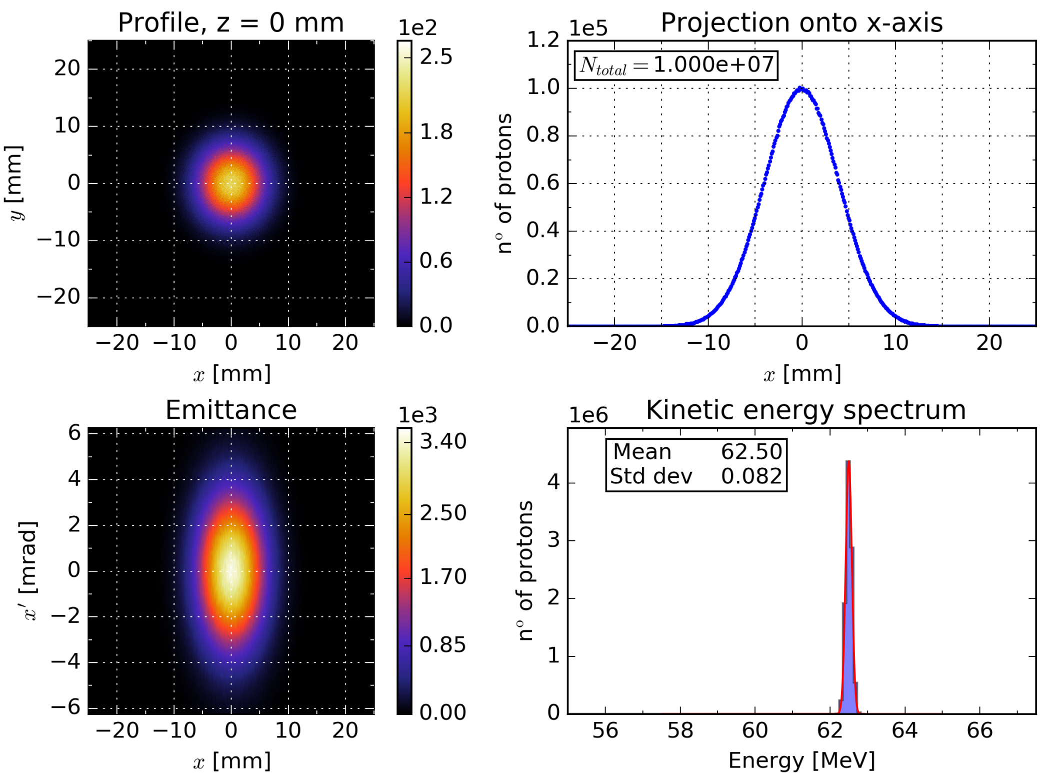

Figure 4.1: **The beam at the source**. The profile plot appears to match the set values of $\sigma_x = 4.0$ mm and $\sigma_y = 4.5$ mm. The energy spectrum is exactly as specified.

stark difference between it and the next plot. At $z = 81$ mm, the protons have just traversed the scattering foil. The emittance plot at this point (Fig. 4.3) indicates that a spread in the angular distribution has occurred. As discussed previously, this should cause the beam profile to spread out once it has travelled further down the beamline. The many scattering events that took place within the thin layer of foil have caused the beam to lose 0.3 MeV in energy which is a large amount considering it had previously lost only 0.01 MeV travelling through roughly 8 cm of air. While advancing further down the line, up until just before the brass stopper ($z =$ 299 mm), the emittance has sheared clockwise, a new halo has formed around the beam profile and its projection in x has formed new tails (Fig. 4.4). These are clear indications that the beam has spread out to some extent. The beam seems to not have lost any energy while the standard deviation in it has decreased and about 40,000 protons were lost. The brass stopper blocks roughly 46% of the remaining beam as can clearly be seen from the ring-like beam profile shown in Fig. 4.5, reducing the beam energy by 0.26 MeV. The second scattering foil sitting immediately behind the brass stopper again leads to a spread in the emittance and is the last element of the dual scattering system. Notice how the emittance now consists of two nearly separate ellipses. From here onwards, there are no major scatterers left in the beamline so the beam continues to spread out and becomes much more uniform.

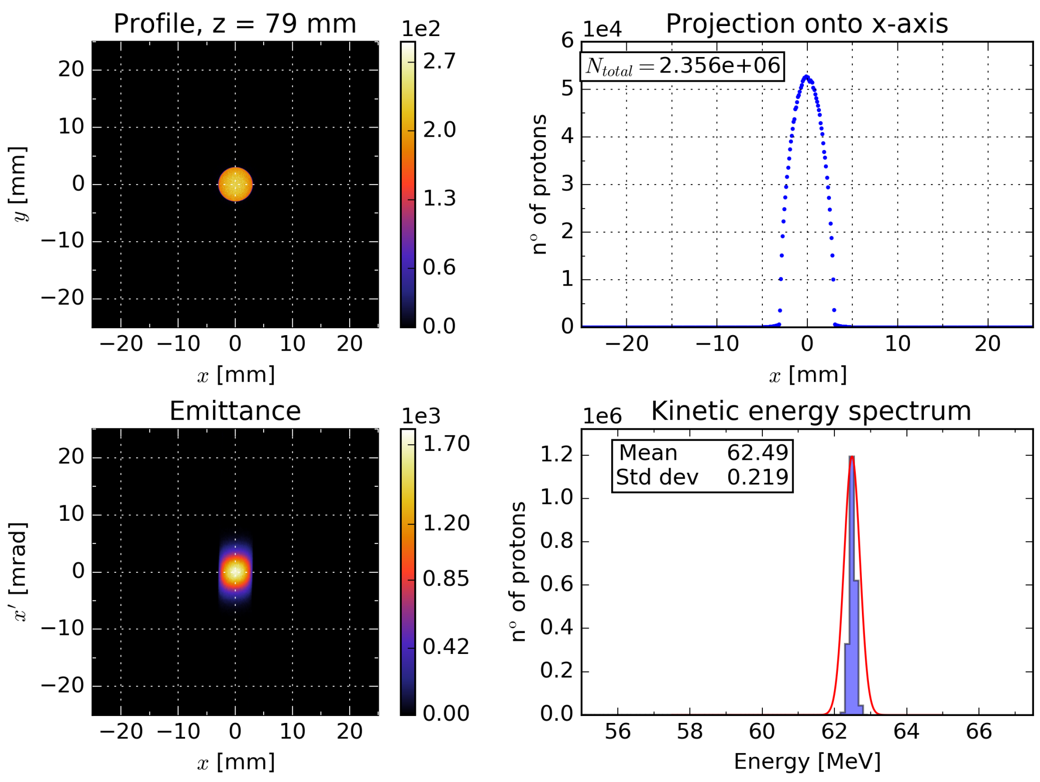

Figure 4.2: **The beam after the first collimator**. A sharp beam of radius 6 mm is left. The emittance has slightly sheared and the beam is starting to lose energy.

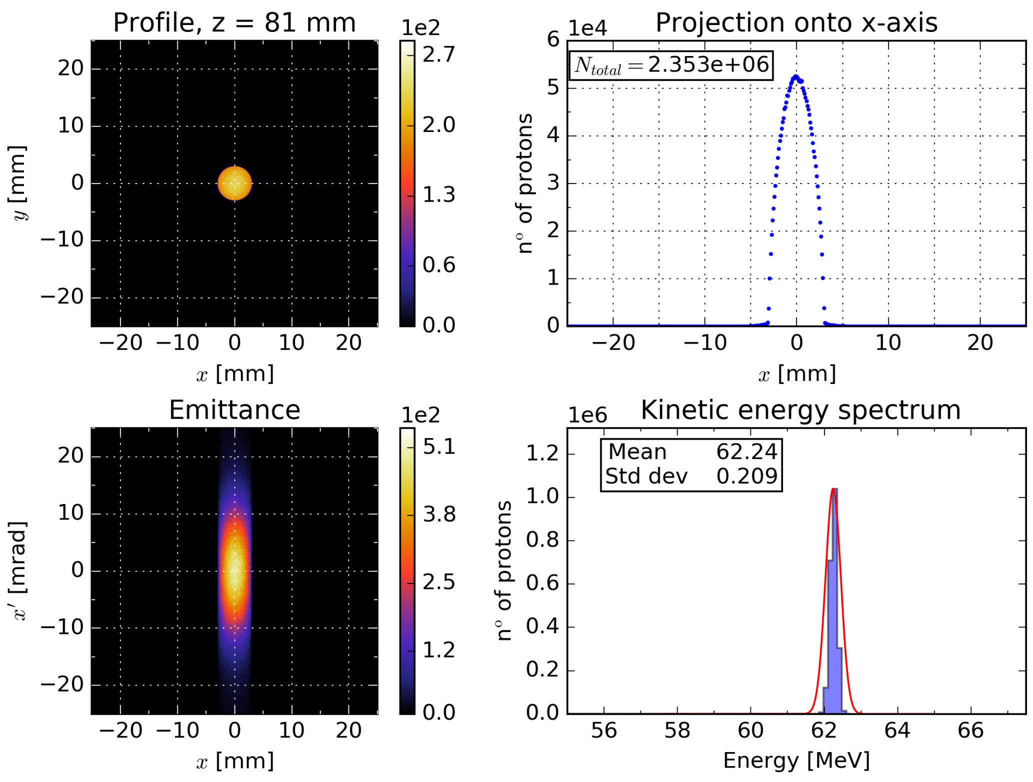

Figure 4.3: **The beam after the first scattering foil**. Although the beam profile is still sharp, the emittance has clearly spread out. The scattering foil has also caused a considerable drop in the beam energy.

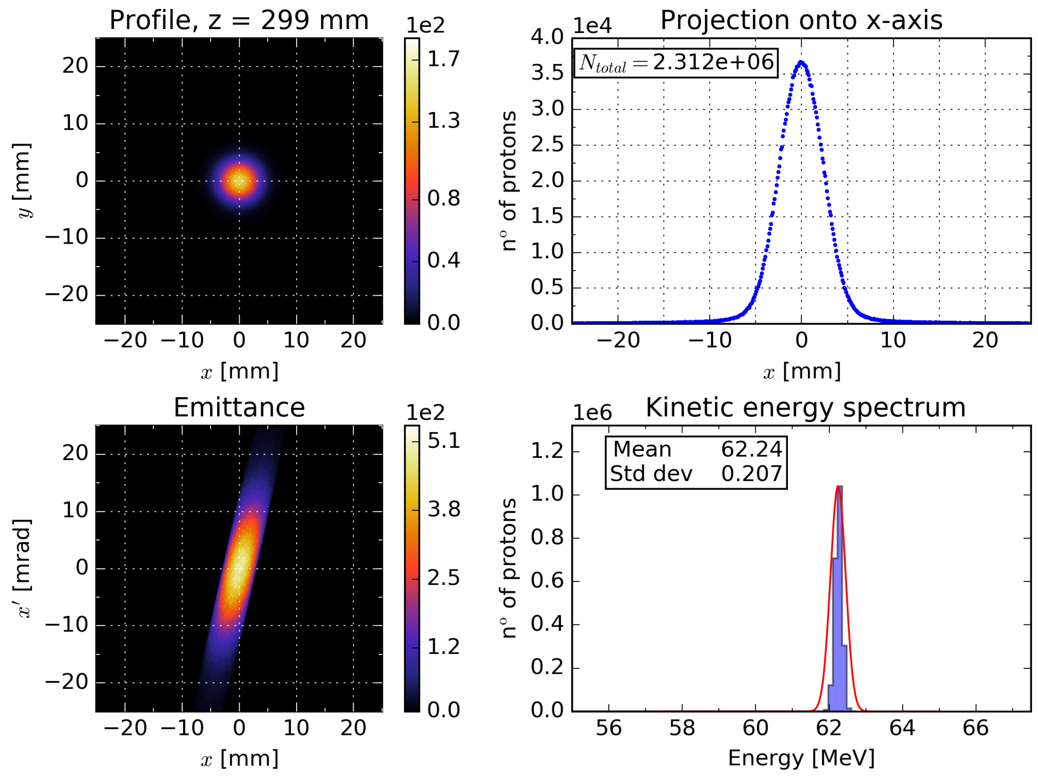

Figure 4.4: **The beam before the brass stopper**. The beam has started to spread out as indicated by the blue halo emerging from the sides of the profile. The emittance is starting to shear.

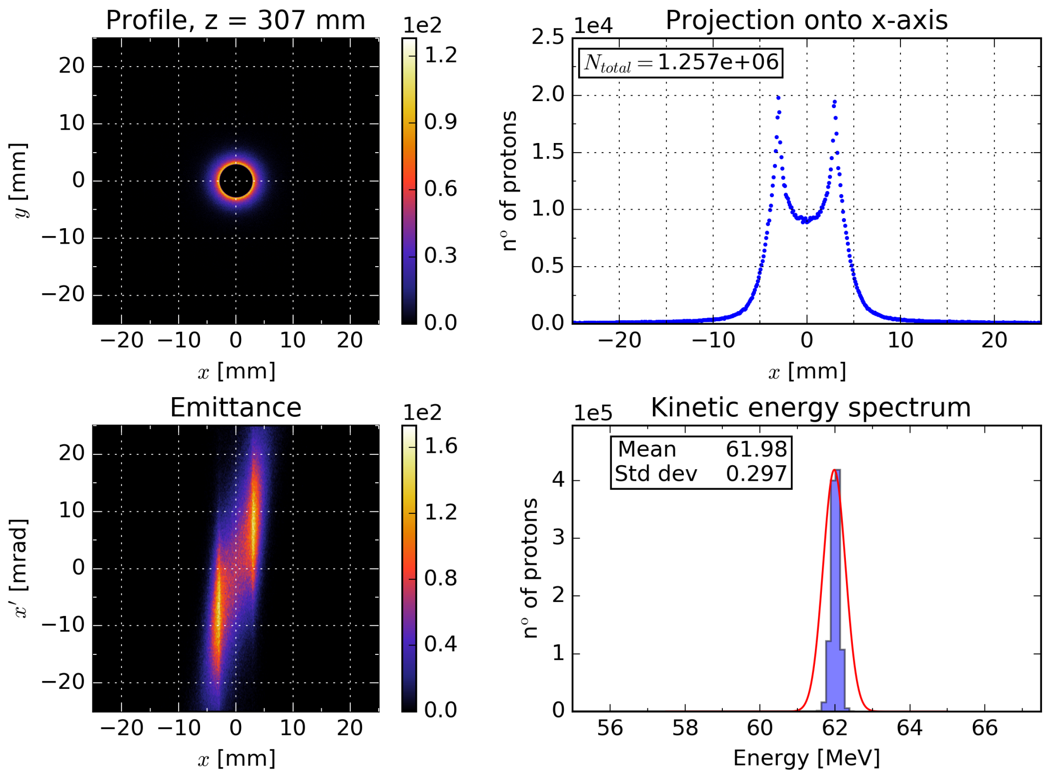

Figure 4.5: **The beam after the brass stopper**. About 47% of the protons were stopped leading to the ring in the profile and drop in beam energy.

A snapshot of the beam just before it enters the dosimetry box is shown in Fig. 4.6, where it now has a diameter of roughly 60 mm—significantly larger than the 6 mm after the first collimator—and the two ellipses in the emittance have nearly merged again. There are only about 960,000 protons left which corresponds to a loss of about 90% of the original beam. The energy lies at 61.03 MeV, a large proportion of which it has yet to lose by scattering off the air in the beamline. Before it is collimated for a final time through the nozzle, the beam is at its greatest extent with a diameter of approximately 80 mm (Fig. 4.7). At an energy of 60.34 MeV, it has lost 0.69 MeV through the second box. It is very uniform up to a radius of about 20 mm, where the number density of protons starts to decrease and its emittance has continued to shear as the beam has continuously diverged. Finally, the halo of low proton density is removed as the beam travels through the nozzle. The figures pictured in Fig. 4.8 where recorded just before the beam entered the detector volume. The beam profile is a disk of high uniformity—there are no discernible differences in the number density across the profile of the beam. Its diameter matches that of the nozzle at 34 mm and its final energy lies at 60.14 MeV.

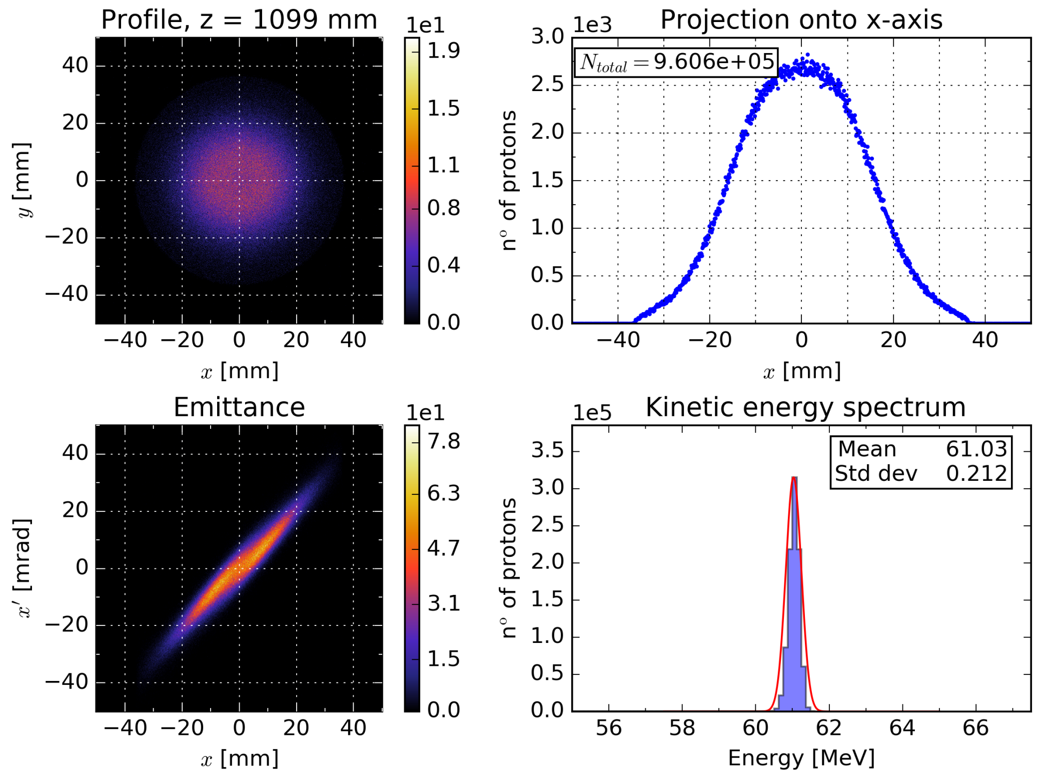

Figure 4.6: **The beam just before the dosimetry box**. A slight ring is still visible in the profile but this example shows how the beam evens out after the dual scattering. The emittance is nearly elliptical again.

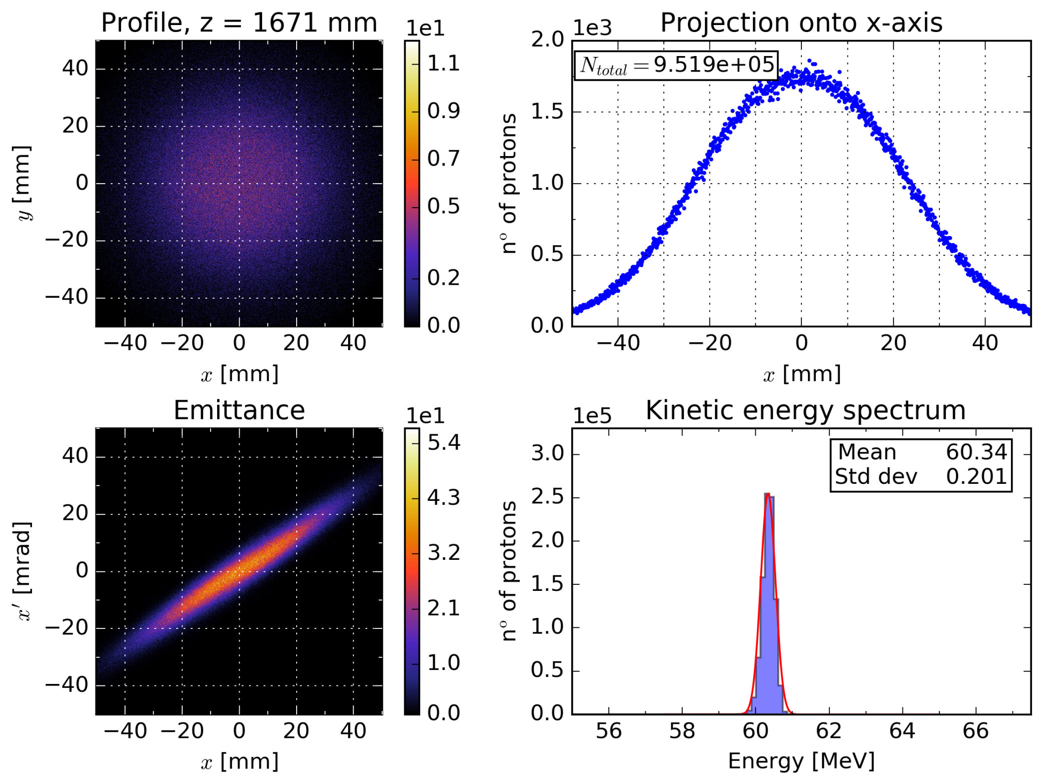

Figure 4.7: **The beam before the nozzle**. The beam is very uniform over a large proportion of its profile and it has lost a lot of energy.

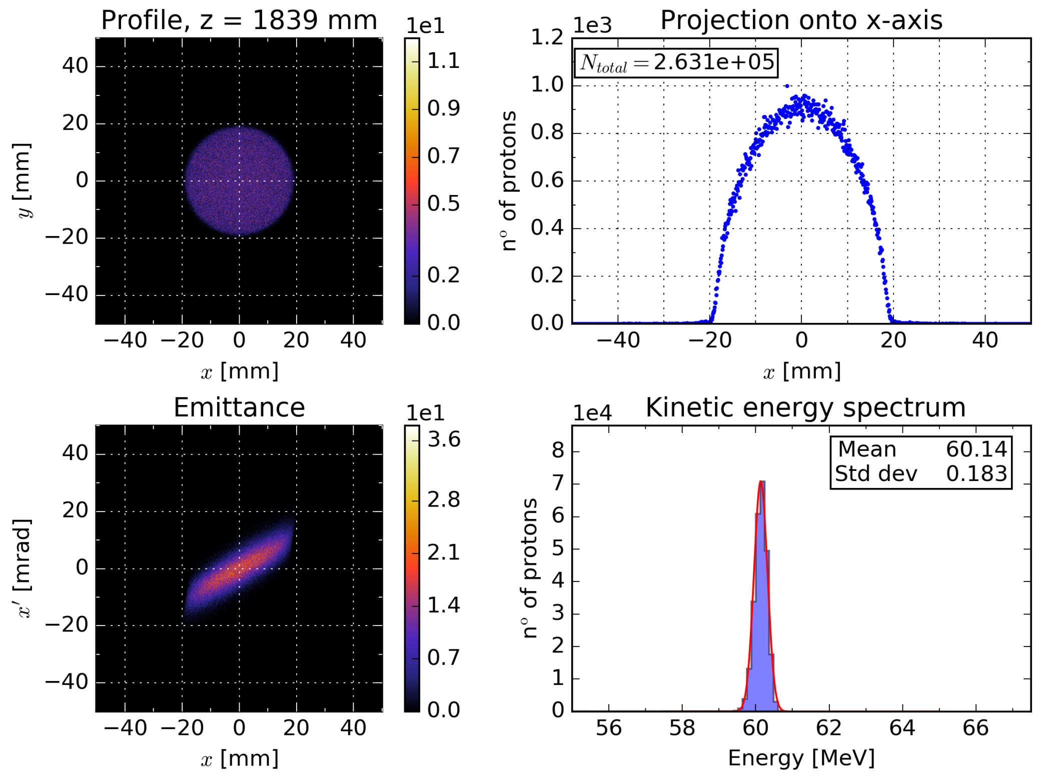

Figure 4.8: **The beam after the nozzle**. A sharp and very uniform beam of diameter 34 mm is formed. The beam has a final energy of 60.14 MeV.

The profile of the dose deposition recorded using a dose deposition scorer in the water volume is shown in Fig. 4.9 below. The Bragg peak is well defined and occurs at a depth of 30.8 mm. The lateral dose deposition was recorded to be compared with the profiles in [27] and [31] and is shown in Fig. 4.10.

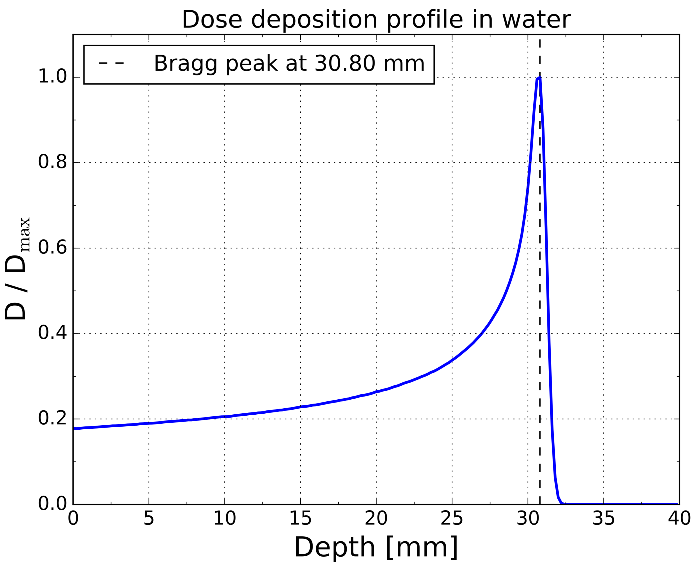

Figure 4.9: **Bragg peak** in the dose deposited in the water volume. The peak occurs at a depth of 30.8 mm. The dose was normalised against the maximum value of the dose deposited.

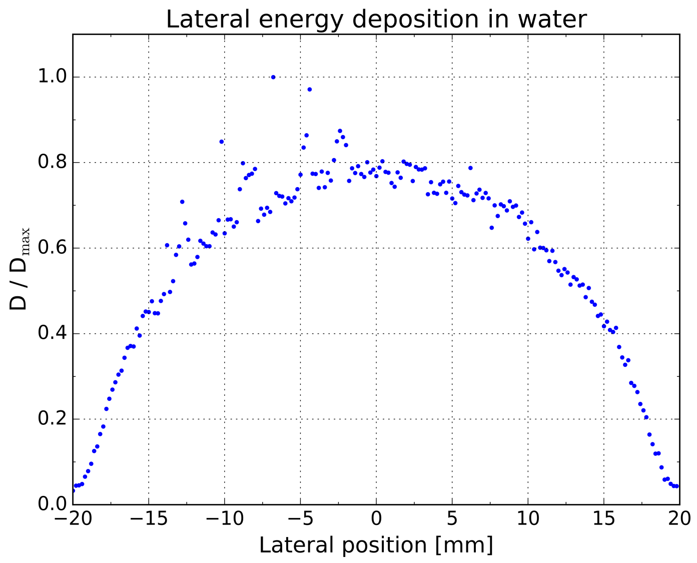

Figure 4.10: **Lateral dose deposition** recorded at the Bragg peak position in the water volume. The dose deposited varies by roughly 25% between −10 mm and 10 mm.

4.3 Discussion

If the simulation behaved as expected, one should be able to roughly read the parameters off the initial beam profile and emittance. Indeed in Fig. 4.1, the width of the plots seems to match the input parameters both in x and y with standard deviations of 4.0 mm and 4.5 mm respectively. The fact that the fit performed on the energy spectrum retrieves the initial energy parameters is also a good indication that the particles are being recorded properly and the fit is performed correctly. The characterisation performed suggests that the beam behaves as expected throughout the beamline. The dual scattering set-up results in a beam profile that is highly uniform with only a very slight ring still visible by the time it reaches the dosimetry box. It travels through the latter undisturbed and a seemingly perfect smooth profile is obtained once it has exited the beamline through the nozzle. Besides this, the emittance exhibits the expected shearing throughout.

The final energy of 60.14 MeV, on the other hand, does not match the expected value of 60.0 MeV quoted in [27]. This could be due to insufficient interactions between the beam and the dosimetry equipment as the layers of aluminium and Mylar in the dose monitors are thin at 1 μm and 5 μm respectively. It is must be pointed out that the expected value of 60.0 MeV might not be correct itself, as there are other inconsistencies in [27]. For instance, the scattering foils are said to have a thickness of 20 μm, when it has been confirmed by the author of the paper himself, that the thickness of 25 μm used in the simulation is indeed correct [32]. Considering this paper was published in 2009, the scattering foils may have simply been replaced in the meantime. Additionally, since a thicker scattering foil should lead to a beam energy lower than 60.0 MeV in the first place, this cannot account for the inaccuracy. Another possibility is that the input energy of 62.5 MeV, along with the other initial parameters, is not accurate. As Clatterbridge has thus far been unable to validate the values used, it seems to be a likely source of error that should be straight forward to investigate and correct for by taking the necessary measurements.

Due to the inaccuracy in the energy of the beam, the penetration depth in the water volume could not be compared directly to that given in [27]. Therefore, the theoretical range for a 60.14 MeV beam had to be calculated using the analytical approximation of the Bragg curve by T. Bortfeld [33]. The resulting plot is shown in Fig. 4.11 below. The peak positions were determined by searching for the maximum value in the corresponding arrays—a method which could be improved upon by fitting to the curves. Here, another aspect of the simulation is confirmed to behave correctly as the peak position obtained in the simulation matches the theoretical value. The difference in the tails is due to the the simulation beam having a spread in its energy, whereas the theoretical curve corresponds is that for a pencil beam.

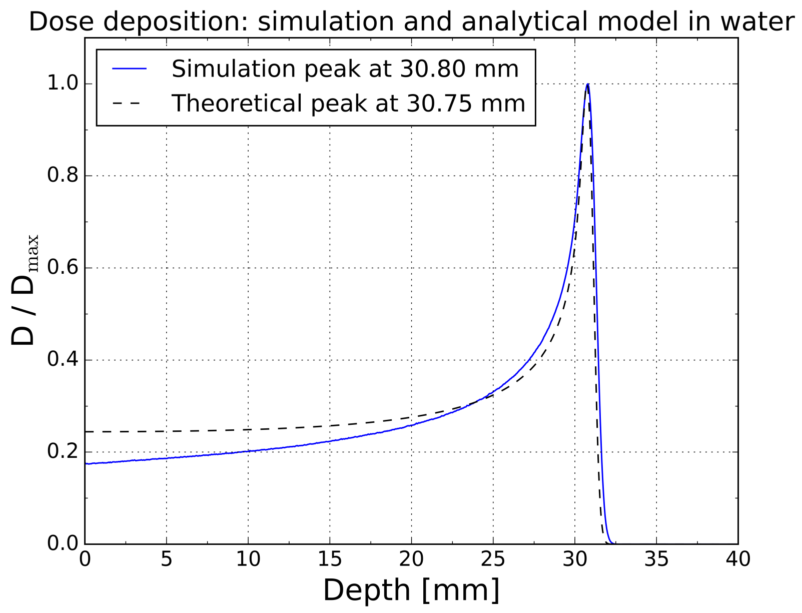

Figure 4.11: **Bragg peak** in the water volume for the simulations and the analytical model for a 60.14 MeV beam. The peak occurs at a depth of 30.80 mm in the simulation and 30.75 mm for the analytical model. The dose was normalised against the maximum value of the dose deposited.

Chapter 5

Conclusion

The simulation behaves as expected in all but one aspect. The final energy of the beam of 60.14 MeV does not quite match the expected value of 60.0 MeV. This is probably due to the input parameters which have so far not been confirmed. Otherwise, all of the following criteria have been considered to have been accurately simulated: the transverse beam profile and its projection onto the x-axis, the beam emittance, and the profile of the dose deposition inside a water phantom.

5.1 Improvements

Firstly, the performance of the simulation could be improved using `G4SensitiveDetector` instances to record the data as any desirable quantity can be accessed through them. In order to use a sensitive detector, it needs to be assigned to a logical volume in the simulation. Empty and arbitrarily thin volumes could be created and have a sensitive detector assigned to them in a loop to record data at many different positions in z. The effects of having a beam travelling through a large number of thin empty volumes would have to be investigated before such a solution is implemented. The assumption is that since the sensitive detectors are optimised for data-passing within the simulation, using them would be less computationally intensive than writing to files manually in the stepping action.

Secondly, as mentioned in Section 4.3, there is a possibility that there are insufficient interactions between the protons in the beam and the layers of aluminium and Mylar leading to the larger-than-expected final energy. It would have to be investigated whether changing the maximum step size to a smaller value improves this behaviour. Alternatively, the thickness of the layers could be increased while decreasing their density to accommodate more steps inside the layer while ensuring likelihood of interactions occurring is the same.

5.2 Next steps

Once the input parameters have been verified and the final energy adjusted, a more advanced model of the detector, possibly even including the PMT, could be incorporated into the simulation. The effect of different rates and intensities on the response of the detector could then be investigated.

Bibliography

[1] Office for National Statistics, "Deaths registered in england and wales: 2014." http://www.ons.gov.uk/peoplepopulationandcommunity/birthsdeathsandmarriages/deaths/bulletins/deathsregistrationsummarytables/2015-07-15#deaths-in-the-uk. [Accessed: 14-08-2016].

[2] Cancer mortality statistics [Online], "Cancer research uk." http://www.cancerresearchuk.org/health-professional/cancer-statistics/mortality. [Accessed: 14-08-2016].

[3] Radiotherapy - NHS Choices [Online]. http://www.nhs.uk/conditions/radiotherapy/Pages/Introduction.aspx. [Accessed: 15-08-2016].

[4] W. C. Röntgen, "Über eine neue art von strahlen," *Ann. d. Phys. u. Chem. N. F.*, vol. 64, no. 1, 1898.

[5] G. M. MacKee, *X-Rays and Radium In the Treatment of Diseases of The skin*, ch. 1. Historical, p. 20. New York, NY: Lea & Febiger, 1921.

[6] Science.hq.nasa.gov [Online], "X-rays." http://science.hq.nasa.gov/kids/imagers/ems/xrays.html. [Accessed: 15-08-2016].

[7] E. C. Halperin, L. W. Brady, C. A. Perez, and D. E. Wazer, *Perez & Brady's Principles and Practice of Radiation Oncology*. Lippincott Williams & Wilkins, 2013.

[8] T. I. Yock and N. J. Tarbell, "Technology insight: proton beam radiotherapy for treatment in pediatric brain tumors," *Nat. Rev. Clin. Oncol.*, vol. 1, no. 2, 2004.

[9] D. Létourneau, et al., "Multileaf collimator performance monitoring and improvement using semiautomated quality control testing and statistical process control," *Med. Phys.*, vol. 41, 2014.

[10] O. K. H. Kobayashi, S. Sakuma and H. Yogo, "Computer-assisted conformation radiotherapy with a variable thickness multi-leaf filter," *Int. J. Radiation Oncology Biol. Phys.*, vol. 16, 1989.

[11] R. Wilson, "Radiological use of fast protons," *Radiology*, vol. 47, no. 5, pp. pp.487–491, 1946.

[12] P. New, "Radiation injury to the nervous system," *Curr. Opin. Neurol.*, vol. 14, pp. pp. 725–734, 2001.

[13] Clatterbridge Cancer Centre: Eye Proton Therapy [Online]. http://www.clatterbridgecc.nhs.uk/professionals/physics-department/cyclotron. [Accessed: 16-08-2016].

[14] UCLH, "Proton beam therapy coming to uclh." https://www.uclh.nhs.uk/News/Pages/ProtonbeamtherapycomingtoUCLH.aspx. [Accessed: 15-08-2016].

[15] Cancer Research UK [Online], "Manchester and london proton beam therapy units confirmed."
http://www.cancerresearchuk.org/about-us/cancer-news/news-report/2013-08-01-manchester-and-london-proton-beam-therapy-units-confirmed. [Accessed: 16-08-2016].

[16] YouTube [Online, "National proton beam therapy programme." https://www.youtube.com/watch?v=2MadsdvYOis. [Accessed: 16-08-2016].

[17] A. Basharina-Freshville, *Search for the neutrinoless double beta decay of 100-Mo with the NEMO3 detector and calorimeter research and development for the SuperNEMO experiment.* PhD thesis, University College London, 2011.

[18] H. F. W. Sadrozinski, "Particle detector applications in medicine," *Nucl. Instr. Meth. Phys. Res A*, vol. 732, pp. pp. 34–39, 2013.

[19] A. Rimoldi, A. Dell'Acqua, "Atlas detector simulation with geant4."
http://atlas-computing.web.cern.ch/atlas-computing/packages/simulation/geant4/geant4.html. [Accessed: 25-03-2017].

[20] Geant4 Collaboration, "Introduction to geant4."
http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/IntroductionToGeant4/html/index.html. [Accessed: 25-03-2017].

[21] Geant4 Collaboration, "5.2. physics processes." http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/ch05s02.html. [Accessed: 25-03-2017].

[22] Geant4 Collaboration, "Chapter 5. tracking and physics." https://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/ch05.html. [Accessed: 25-03-2017].

[23] Ben Krikler, "General paricle source control commands.." http://www.hep.ph.ic.ac.uk/~bek07/comet/SimG4/macro_commands/_gps_.html. [Accessed: 25-03-2017].

[24] Geant4 Collaboration, "8. geant4 material database." http://geant4.web.cern.ch/geant4/workAreaUserDocKA/Backup/Docbook_UsersGuides_beta/ForApplicationDeveloper/html/apas08.html. [Accessed: 25-03-2017].

[25] Geant4 Collaboration, "Scoring quantity of the mesh.." https://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/AllResources/Control/UIcommands/_score_quantity_.html. [Accessed: 25-03-2017].

[26] Geant4 Collaboration, "8.3 the visualization drivers." http://geant4.cern.ch/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Visualization/visdrivers.html#DAWN. [Accessed: 25-03-2017].

[27] A. Kacperek, "Protontherapy of eye tumours in the uk: A review of treatment at clatterbridge," *Appl. Radiat. Isot*, vol. 67, p. 378–386, 2009.

[28] R. Zhang & W. D. Newhauser, "Calculation of water equivalent thickness of materials of arbitrary density, elemental composition and thickness in proton beam irradiation," *Phys. Med. Biol.*, vol. 54, p. 1383–1395, 2009.

[29] G. Radegran & B. Saltin, "Human femoral artery diameter in relation to knee extensor muscle mass, peak blood flow, and oxygen uptake," *Am. J. Physiol. Heart Circ. Physiol.*, vol. 278, pp. 162–167, 2000.

[30] M. P. Stockli, "Measuring and analyzing the transverse emittance of charged particle beams," *Am. J. Physiol. Heart Circ. Physiol.*, vol. 278, pp. 162–167, 2000.

[31] G. A. P. Cirrone, et al., "A 62-mev proton beam for the treatment of ocular melanoma at laboratori nazionali del sud-infn," *IEEE Trans. Nucl. Sci.*, vol. 51, pp. 860–865, 2004.

[32] A. Kacperek, Private communication, 2017.

[33] T. Bortfeld, "An analytical approximation of the bragg curve for therapeutic proton beams," *Med. Phys.*, vol. 24, p. 2024–2033, 1997.

Chapter 6

Appendices

6.1 Running simultaneous simulations

All simulations were executed on the HEP Linux cluster at plus1.hep.ucl.ac.uk. It runs Scientific Linux 6.8 and uses the PBS job scheduler. Simulations were run in GEANT4 v10.2.1 with the CLHEP v2.3.1.1 extension and visualised in DAWN v3.90b. A simulation of 1 million protons tracked in 20 different locations can take anywhere between 8 and 24 hours to complete, depending on the node it runs on. As this is a severe limitation, 100 simulations were run simultaneously, all submitted using `qsub`. The data were subsequently combined into a single data set using a python script. Ordinarily, the `proton.mac` file shown in Section 6.2.1 below would be used to run a single simulation. When multiple are run consecutively, the `proton.mac_template` file (Section 6.2.2) is used. The main difference lies in the fact the two seeds must be set for each individual simulation in a repeatable fashion. This is achieved by replacing them with variables and assigning values to these when the simulations are submitted using the script shown in Section 6.3.1. For consistency the values normally used are the run ID and the run ID + 100. Once the simulations have completed, they can be combined using a script. It is usually helpful to compress the directory into a `tar.gz` file before downloading it from the cluster as naturally a smaller file size reduces download speed, but downloading a single file is also faster than downloading multiple files of the same total size.

6.2 Macros

6.2.1 proton.mac

```
# proton.mac

# set verbosity for debugging etc.
/control/verbose 2
/run/verbose 0
/tracking/verbose 0
/run/particle/verbose 0
#/run/particle/dumpList

# set detector parameters
/protonB/det/setMat Water
/protonB/det/position 0. 0. -2340 mm
/protonB/det/setSizeXY 40 mm
/protonB/det/setSizeZ  40 mm
/protonB/det/setSliceSizeXY 40 mm
/protonB/det/sliceNumber 40

# tracking: All, None or distance in z
/steppingAction/tracking None

#### initialise scorers ####
/control/execute score_init.mac

# choose physics list
/protonB/phys/addPhysics QGSP_BIC_EMY

# production thresholds (recommended cut off not bigger than 10% of slice thickness)
/protonB/phys/setCuts 0.2 mm

/run/initialize
/random/setSeeds 1470232172 1402668138

#### visualisation ####
/control/execute visualisation.mac

# gps settings
/gps/particle proton
/gps/number 1 # per event

/gps/ene/type Gauss
/gps/ene/mono 62.5 MeV
/gps/ene/sigma 0.082 MeV

/gps/position 0.0 0.0 -4200 mm
/gps/pos/type Beam
/gps/pos/sigma_x 4.0 mm
/gps/pos/sigma_y 4.5 mm

/gps/ang/type beam2d
/gps/ang/sigma_x 2.3 mrad
/gps/ang/sigma_y 1.2 mrad
/gps/ang/rot1 -1 0 0

# step limit (not bigger than 5% of slice thickness)
/protonB/stepMax 0.1 mm

# print every N steps
/protonB/event/printModulo 100

# number of events
/run/beamOn 100000

# dump scores to a file
/control/execute score_dump.mac

```

6.2.2 proton.mac_template

```
# proton.mac_template

# set verbosity for debugging etc.
/control/verbose 2
/run/verbose 0
/tracking/verbose 0
/run/particle/verbose 0
#/run/particle/dumpList

# set geometry
/protonB/det/setMat Water
/protonB/det/position 0. 0. -2340. mm
/protonB/det/setSizeXY 40 mm
/protonB/det/setSizeZ  40 mm
/protonB/det/setSliceSizeXY 40 mm
/protonB/det/sliceNumber 40

# tracking: All, None or distance in z
/steppingAction/tracking None

#### scoring ####
/control/execute ../../score_init.mac

# set physics list
/protonB/phys/addPhysics QGSP_BIC_EMY

# production thresholds
/protonB/phys/setCuts 0.2 mm


/run/initialize
# these placeholders will be replaced once the simulations are submitted
/random/setSeeds ${seed1} ${seed2}

# gps settings
/gps/particle proton
/gps/number 1

/gps/ene/type Gauss
/gps/ene/mono 62.5 MeV      # particle energy used in Clatterbridge Centre
/gps/ene/sigma 0.082 MeV

/gps/position 0.0 0.0 -4200 mm
/gps/pos/type Beam
/gps/pos/sigma_x 4.0 mm
/gps/pos/sigma_y 4.5 mm

/gps/ang/type beam2d
/gps/ang/sigma_x 2.3 mrad
/gps/ang/sigma_y 1.2 mrad
/gps/ang/rot1 -1 0 0

# step limit
/protonB/stepMax 0.1 mm

# print every N steps
/protonB/event/printModulo 100

# number of events
/run/beamOn 100000

# dump scores to a file
/control/execute ../../score_dump.mac

```

6.2.3 visualisation.mac

```
# visualisation macro
# create .prim files suitable for viewing in DAWN:
/vis/open DAWNFILE

# disable auto refresh and quieten vis messages whilst scene and
# trajectories are established
/vis/viewer/set/autoRefresh false
/vis/verbose errors

# draw geometry (can specifiy volumes here, e.g. 'AlBox2')
/vis/drawVolume

# specify style (surface s or wireframe w):
/vis/viewer/set/style s

# specify zoom value and view angle
/vis/viewer/zoom 1.45
/vis/viewer/set/viewpointThetaPhi 55. 20. # my favourite view

# draw coordinate axes
#/vis/scene/add/axes 0 0 0 1 cm
/vis/scene/add/trajectories

# draw geometry before events
/vis/viewer/flush

# draw smooth trajectories at end of event, showing trajectory points
# as markers 2 pixels wide
/vis/scene/add/trajectories smooth

# only show protons (for clarity: good for reports)
#/vis/filtering/trajectories/create/particleFilter
#/vis/filtering/trajectories/particleFilter-0/add proton

# select colour by particle ID
/vis/modeling/trajectories/create/drawByParticleID
/vis/modeling/trajectories/drawByParticleID-0/set e- red
/vis/modeling/trajectories/drawByParticleID-0/set e+ cyan
/vis/modeling/trajectories/drawByParticleID-0/set proton blue
/vis/modeling/trajectories/drawByParticleID-0/set gamma green
/vis/modeling/trajectories/drawByParticleID-0/set neutron yellow
/vis/modeling/trajectories/drawByParticleID-0/set pi+ magenta
/vis/modeling/trajectories/drawByParticleID-0/set pi- magenta
/vis/modeling/trajectories/drawByParticleID-0/set pi0 magenta
/vis/modeling/trajectories/drawByParticleID-0/set muon black
/vis/modeling/trajectories/drawByParticleID-0/set opticalphoton brown

# to superimpose all of the events from a given run
/vis/scene/endOfEventAction accumulate

# Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/verbose warnings

```

6.2.4 score_init.mac

```
# isocentre at z = 1829 mm
/score/create/boxMesh isoEnergyLat
/score/mesh/boxSize 20. 20. 0.5 mm
/score/mesh/nBin 200 1 1
/score/mesh/translate/xyz 0. 0. -2371 mm
/score/quantity/energyDeposit energyDeposit
/score/quantity/doseDeposit doseDeposit
/score/close

# middle of detector at z = 1860 mm
/score/create/boxMesh detEnergyLat
/score/mesh/boxSize 20. 20. 0.5 mm
/score/mesh/nBin 200 1 1
/score/mesh/translate/xyz 0. 0. -2340 mm
/score/quantity/energyDeposit energyDeposit
/score/quantity/doseDeposit doseDeposit
/score/close

# detector at z = 1860 mm
/score/create/boxMesh detEnergyLon
/score/mesh/boxSize 20. 20. 20. mm
/score/mesh/nBin 1 1 800
/score/mesh/translate/xyz 0. 0. -2340 mm
/score/quantity/energyDeposit energyDeposit
/score/quantity/doseDeposit doseDeposit
/score/close

/score/create/boxMesh detFluxLon
/score/mesh/boxSize 20. 20. 20. mm
/score/mesh/nBin 1 1 800
/score/mesh/translate/xyz 0. 0. -2340 mm
/score/quantity/flatSurfaceFlux protonFlux 1
/score/filter/particle protonFilter proton
/score/close

# Bragg peak at z = 1870.8 mm in water (30.8 mm depth)
/score/create/boxMesh braggEnergyLat
/score/mesh/boxSize 20. 20. 0.5 mm
/score/mesh/nBin 200 1 1
/score/mesh/translate/xyz 0. 0. -2329.2 mm
/score/quantity/energyDeposit energyDeposit
/score/quantity/doseDeposit doseDeposit
/score/close

```

6.2.5 score_dump.mac

```
/score/dumpQuantityToFile isoEnergyLat energyDeposit IsoEnergyDepLat.txt
/score/dumpQuantityToFile isoEnergyLat doseDeposit   IsoDoseDepLat.txt
/score/dumpQuantityToFile detEnergyLat energyDeposit DetEnergyDepLat.txt
/score/dumpQuantityToFile detEnergyLat doseDeposit   DetDoseDepLat.txt
/score/dumpQuantityToFile detEnergyLon energyDeposit DetEnergyDepLon.txt
/score/dumpQuantityToFile detEnergyLon doseDeposit   DetDoseDepLon.txt
/score/dumpQuantityToFile detFluxLon   protonFlux    DetFluxLon.txt
/score/dumpQuantityToFile braggEnergyLat energyDeposit BraggEnergyDepLat.txt
/score/dumpQuantityToFile braggEnergyLat doseDeposit   BraggDoseDepLat.txt
```

6.3 Bash scripts

6.3.1 submit.sh

```bash
#!/bin/bash

# check if sl6 env is set
env=$(printenv | grep "PBTENV")
if [[ "$env" != "PBTENV=SL6 Production" ]]; then
    echo "  Need to set SL6 Production environment."
    exit
fi

# check if arguments were given
if [[ -z $1 ]]; then
    echo "  No arguments supplied. Need to specify the number of simulations required."
    exit
fi

# get the number of events from the proton.mac_template
NEVENTS=$(grep '/run/beamOn' ../proton.mac_template | sed 's/[^0-9]*//g')

# get number of simulations and queue from arguments
NSIMULATIONS=$1
QUEUE_TYPE=$2

echo "  Submitting $NSIMULATIONS simulations of $NEVENTS events..."
for ((i = 1; i <= $NSIMULATIONS; i++)); do

    # replace placeholders with random seeds
    cat ../beam_1M_temp | sed -e "s/\${i}/$i/" >> beam_1M_s$i
    qsub -q $QUEUE_TYPE beam_1M_s$i
    rm beam_1M_s$i
done
```

6.4 Geant4 class files

6.4.1 DetectorConstruction.cc

```cpp
// M Hentz, 2016
// This class contains the geometry definition of the simulation.

#include "DetectorConstruction.hh"
#include "DetectorMessenger.hh"
#include "G4Material.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4UniformMagField.hh"
#include "G4PVReplica.hh"
#include "G4SubtractionSolid.hh"
#include "G4Transform3D.hh"
#include "G4GeometryManager.hh"
#include "G4PhysicalVolumeStore.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4SolidStore.hh"
#include "G4VisAttributes.hh"
#include "G4NistManager.hh"
#include "G4UnitsTable.hh"
#include "G4TransportationManager.hh"
#include "G4RunManager.hh"
#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"

DetectorConstruction::DetectorConstruction()
: G4VUserDetectorConstruction(),
fWorldMaterial(0),
fAbsorMaterial(0),
fDetectorMessenger(0)
{
  // set default values
  fAbsorSizeZ = fAbsorSizeXY = 40*CLHEP::mm;

  detSizeXY = fAbsorSizeXY + 5*CLHEP::mm;
  detSizeZ  = fAbsorSizeZ  + 5*CLHEP::mm;

  fWorldSizeX = fWorldSizeZ = 9450*CLHEP::mm;
  fWorldSizeY = 4450*CLHEP::mm;

  fLayerSizeXY = 20*CLHEP::mm;
  fLayerSizeZ = 2*CLHEP::mm;

  DefineMaterials();

  // create commands for interactive definition of the detector
  fDetectorMessenger = new DetectorMessenger(this);
}


DetectorConstruction::~DetectorConstruction()
{ delete fDetectorMessenger;}


G4VPhysicalVolume* DetectorConstruction::Construct()
{
  return ConstructVolumes();
}


void DetectorConstruction::DefineMaterials()
{
  // nist manager to get materials from database
  G4NistManager* nistMan = G4NistManager::Instance();

  // --- define Elements ---
  G4double z, a;

  G4Element* H  = new G4Element("Hydrogen" , "H" , z=1. , a=1.008*g/mole);
  G4Element* Li = new G4Element("Lithium"  , "Li", z=3. , a=6.941*g/mole);
  G4Element* B = new G4Element("Boron"     , "B" , z=5. , a=10.81*g/mole);
```

```
G4Element* C  = new G4Element("Carbon"    , "C" , z=6. , a=12.01*g/mole);
G4Element* N  = new G4Element("Nitrogen"  , "N" , z=7. , a=14.01*g/mole);
G4Element* O  = new G4Element("Oxygen"    , "O" , z=8. , a=16.00*g/mole);
G4Element* Na = new G4Element("Sodium"    , "Na", z=11., a=22.99*g/mole);
G4Element* Mg = new G4Element("Magnesium", "Mg", z=12., a=24.305*g/mole);
G4Element* Al = new G4Element("Aluminium", "Al", z=13., a=26.98*g/mole);
G4Element* Si = new G4Element("Silicon"   , "Si", z=14., a=28.09*g/mole);
G4Element* Ar = new G4Element("Argon"     , "Ar", z=18., a=39.948*g/mole);
G4Element* Ca = new G4Element("Calcium"   , "Ca", z=20., a=40.08*g/mole);
G4Element* Fe = new G4Element("Iron"      , "Fe", z=26., a=55.85*g/mole);
G4Element* Co = new G4Element("Cobalt"    , "Co", z=27., a=58.933*g/mole);
G4Element* Cu = new G4Element("Copper"    , "Cu", z=29., a=63.546*g/mole);
G4Element* Zn = new G4Element("Zinc"      , "Zn", z=30., a=65.38*g/mole);
G4Element* Eu = new G4Element("Europium"  , "Eu", z=63., a=151.964*g/mole);
G4Element* Cs = new G4Element("Caesium"   , "Cs", z=55., a=132.905*g/mole);
G4Element* W  = new G4Element("Tungsten"  , "W" , z=74., a=183.84*g/mole);


// --- define materials ---
G4double density, temperature, pressure, fractionmass;
G4int natoms, ncomponents;

// vacuum
density     = universe_mean_density;    //from PhysicalConstants.h
pressure    = 3.e-18*pascal;
temperature = 2.73*kelvin;
vacuum = new G4Material("Vacuum", z=1, a=1.008*g/mole,
                        density, kStateGas, temperature, pressure);

// air
G4Material* air = new G4Material("Air", density=1.290*mg/cm3, ncomponents=3);
air->AddElement( N,  fractionmass=0.7810 );       // 78.10%
air->AddElement( O,  fractionmass=0.2096 );       // 20.96%
air->AddElement( Ar, fractionmass=0.0094 );       // 0.94%

// water
G4Material* H2O = new G4Material("Water", density= 1.0*g/cm3,
                                 ncomponents=2);
H2O->AddElement(H, natoms=2);
H2O->AddElement(O, natoms=1);
H2O->GetIonisation()->SetMeanExcitationEnergy(78.0*eV);

// PVT scintillator
G4Material* scintillatorPVT = new G4Material("Scintillator_PVT",
                                              density=1.023*g/cm3, ncomponents=2);
scintillatorPVT->AddElement( C, natoms=9 );
scintillatorPVT->AddElement( H, natoms=10 );

// Concrete
G4Material* marbleConcrete = new G4Material("MarbleConcrete",
                                              density=2.7*g/cm3, ncomponents=12);
marbleConcrete->AddElement( Ca, fractionmass=0.473942 );    // 47.3942%
marbleConcrete->AddElement( O,  fractionmass=0.375    );    // 37.5%
marbleConcrete->AddElement( C,  fractionmass=0.105    );    // 10.5%
marbleConcrete->AddElement( Si, fractionmass=0.024    );    // 2.4%
marbleConcrete->AddElement( Fe, fractionmass=0.01     );    // 1%
marbleConcrete->AddElement( Al, fractionmass=0.007    );    // 0.7%
marbleConcrete->AddElement( Mg, fractionmass=0.003    );    // 0.3%
marbleConcrete->AddElement( Na, fractionmass=0.002    );    // 0.2%
marbleConcrete->AddElement( Li, fractionmass=0.000037 );    // 37 ppm
marbleConcrete->AddElement( Co, fractionmass=0.000018 );    // 18 ppm
marbleConcrete->AddElement( Cs, fractionmass=0.000002 );    // 2 ppm
marbleConcrete->AddElement( Eu, fractionmass=0.000001 );    // 1 ppm

// Aluminium
G4Material* Al_mat = new G4Material("Aluminium", density=2.7*g/cm3, ncomponents=1);
Al_mat->AddElement( Al, fractionmass=1 );

// Brass
brass = new G4Material("Brass", density=8.75*g/cm3, ncomponents=2);
brass->AddElement( Cu, fractionmass=0.7 );
brass->AddElement( Zn, fractionmass=0.3 );
```

```

  // Tungsten
  tungsten = new G4Material("Tungsten", density=19.25*g/cm3, ncomponents=1);
  tungsten->AddElement( W, fractionmass=1. );

  // Kapton
  kapton = new G4Material("Kapton", density=1.42*g/cm3, ncomponents=4);
  kapton->AddElement( H, fractionmass=0.027 );
  kapton->AddElement( C, fractionmass=0.691 );
  kapton->AddElement( N, fractionmass=0.073 );
  kapton->AddElement( O, fractionmass=0.209 );

  // Iron
  iron = new G4Material("Iron", density=7.874*g/cm3, ncomponents=1);
  iron->AddElement( Fe, fractionmass=1. );

  // PMMA
  PMMA = new G4Material("PMMA", density=1.18*g/cm3, ncomponents=3);
  PMMA->AddElement( C, natoms = 5 );
  PMMA->AddElement( O, natoms = 2 );
  PMMA->AddElement( H, natoms = 8 );

  // Mylar
  mylar = new G4Material("Mylar", density=1.397*g/cm3, ncomponents=3);
  mylar->AddElement( C, natoms= 10 );
  mylar->AddElement( H, natoms=  8 );
  mylar->AddElement( O, natoms=  4 );

  //Borated plastic (5% borated polyethylene)
  boratedPlastic = new G4Material("BoratedPlastic", density=1.04*g/cm3, ncomponents = 3);
  boratedPlastic->AddElement( B, fractionmass=0.05  );
  boratedPlastic->AddElement( C, fractionmass=0.317 );
  boratedPlastic->AddElement( H, fractionmass=0.633 );

  // Stainless steel - using nist manager
  steel = nistMan->FindOrBuildMaterial("G4_STAINLESS-STEEL");

  // --- list materials ---
  airMaterial      = air;
  waterMaterial    = H2O;
  scintillatorMaterial = scintillatorPVT;
  marbleMaterial   = marbleConcrete;
  aluminium        = Al_mat;

  // --- default materials ---
  fWorldMaterial = airMaterial;
  fAbsorMaterial = scintillatorMaterial;

}


G4VPhysicalVolume* DetectorConstruction::ConstructVolumes()
{
  G4GeometryManager::GetInstance()->OpenGeometry();
  G4PhysicalVolumeStore::GetInstance()->Clean();
  G4LogicalVolumeStore::GetInstance()->Clean();
  G4SolidStore::GetInstance()->Clean();

  // used to check if volumes overlap
  G4bool checkOverlaps = true;

  // defining these here makes definitions of G4Tubs easier later
  G4double startAngle = 0;
  G4double spanningAngle = 2*CLHEP::pi;


  // --- World ---
  G4Box* sWorld = new G4Box("World",
                              9450*CLHEP::mm/2,4450*CLHEP::mm/2, 9450*CLHEP::mm/2);
  G4LogicalVolume* lWorld = new G4LogicalVolume(sWorld, fWorldMaterial, "World");
  G4VPhysicalVolume* pWorld = new G4PVPlacement(0,
                                  G4ThreeVector(), lWorld, "World", 0, false, 0);
```

```


// Simple concrete box - emulates scattering from ceiling, floor, walls.
sRoomOut = new G4Box("Walls", 8150*CLHEP::mm/2, 3550*CLHEP::mm/2, 8550*CLHEP::mm/2);
lRoomOut = new G4LogicalVolume(sRoomOut, marbleMaterial, "Walls");
pRoomOut = new G4PVPlacement(0, G4ThreeVector(), lRoomOut, "Walls",
                            lWorld, false, 0, checkOverlaps);


// --- Air-filled inner room ---
sRoomIn = new G4Box("Inner room", 8000*CLHEP::mm/2, 3400*CLHEP::mm/2, 8400*CLHEP::mm/2);
lRoomIn = new G4LogicalVolume(sRoomIn, airMaterial, "Inner room");
pRoomIn = new G4PVPlacement(0, G4ThreeVector(), lRoomIn, "Inner room",
                            lRoomOut, false, 0, checkOverlaps);



// --->>> BEAM LINE <<<--- //


// 1st section - Aluminium tube

// - 1st Aluminium tube - touching the wall
//        -> z position = -4200 + (356/2)

// -- outer cylinder (solid aluminium) ---
sAlTube1Out = new G4Tubs("AlTube1Out", 0., 38*CLHEP::mm, 356*CLHEP::mm/2,
                        startAngle, spanningAngle);
lAlTube1Out = new G4LogicalVolume(sAlTube1Out, aluminium, "AlTube1Out");
pAlTube1Out = new G4PVPlacement(0, G4ThreeVector(0., 0., -4022*CLHEP::mm),
                  lAlTube1Out, "AlTube1Out", lRoomIn,false, 0, checkOverlaps);


// -- inner cylinder (vacuum) ---
sAlTube1In = new G4Tubs("AlTube1In", 0, 36*CLHEP::mm, 356*CLHEP::mm/2,
                        startAngle, spanningAngle);
lAlTube1In = new G4LogicalVolume(sAlTube1In, vacuum, "AlTube1In");
pAlTube1In = new G4PVPlacement(0,G4ThreeVector(), lAlTube1In, "AlTube1In",
                               lAlTube1Out, false, 0, checkOverlaps);


// --- 1st collimator ---
sCollim1 = new G4Tubs("Collimator1", 3*CLHEP::mm, 36*CLHEP::mm, 10*CLHEP::mm/2,
                      startAngle, spanningAngle);
lCollim1 = new G4LogicalVolume(sCollim1,brass,"Collimator1");
pCollim1 = new G4PVPlacement(0,G4ThreeVector(0, 0, -123*CLHEP::mm),
                  lCollim1, "Collimator1", lAlTube1In, false, 0, checkOverlaps);


// --- 1st scatter foil ---
sScatterFoil1 = new G4Tubs("ScatterFoil1", 0, 36*CLHEP::mm, 0.025*CLHEP::mm/2,
                          startAngle, spanningAngle);
lScatterFoil1 = new G4LogicalVolume(sScatterFoil1, tungsten, "ScatterFoil1");
pScatterFoil1 = new G4PVPlacement(0, G4ThreeVector(0,0,-97.9875*CLHEP::mm),
                  lScatterFoil1, "ScatterFoil1", lAlTube1In, false, 0, checkOverlaps);


// --- beam stopper ---
sStopper = new G4Tubs("Stopper", 0, 2.855*CLHEP::mm, 6.6*CLHEP::mm/2,
                      startAngle, spanningAngle);
lStopper= new G4LogicalVolume(sStopper,brass,"Stopper");
pStopper = new G4PVPlacement(0, G4ThreeVector(0, 0, 125.3*CLHEP::mm),
                      lStopper, "Stopper", lAlTube1In, false, 0, checkOverlaps);


// --- 2nd scatter foil ---
pScatterFoil2 = new G4PVPlacement(0,G4ThreeVector(0,0,128.6125*CLHEP::mm),
                  lScatterFoil1, "ScatterFoil2", lAlTube1In, false, 0, checkOverlaps);




```

```
// --- Kapton window ---
sKaptonWindow = new G4Tubs("Kapton", 0, 36*CLHEP::mm, 0.05*CLHEP::mm/2,
                          startAngle, spanningAngle);
lKaptonWindow = new G4LogicalVolume(sKaptonWindow, kapton, "Kapton");
pKaptonWindow = new G4PVPlacement(0,G4ThreeVector(0,0,-3843.975*mm),
                      lKaptonWindow, "Kapton", lRoomIn, false, 0, checkOverlaps);



// 2nd section - Aluminium box

// --- 1st Aluminium box ---
// -- use subtraction solid -- filled with air
sAlBox1Out = new G4Box("AlBox1Out", 200*CLHEP::mm/2, 200*CLHEP::mm/2, 200*CLHEP::mm/2);
sAlBox1In = new G4Box("AlBox1In", 160*CLHEP::mm/2, 160*CLHEP::mm/2, 160*CLHEP::mm/2);

sAlBox1 = new G4SubtractionSolid("AlBox1", sAlBox1Out, sAlBox1In);
lAlBox1 = new G4LogicalVolume(sAlBox1, aluminium, "AlBox1");
pAlBox1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -3650*CLHEP::mm),
                  lAlBox1, "AlBox1", lRoomIn, false, 0, checkOverlaps);


// -- holes in box for beamline
sAlBox1Hole1 = new G4Tubs("AlBox1Hole1", 0, 36*CLHEP::mm, 20*CLHEP::mm/2,
                          startAngle, spanningAngle);
lAlBox1Hole1 = new G4LogicalVolume(sAlBox1Hole1, airMaterial, "AlBox1Hole1");
pAlBox1Hole1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -90*CLHEP::mm),
                      lAlBox1Hole1, "AlBox1Hole1", lAlBox1, false, 0, checkOverlaps);

// place another copy on the other side
pAlBox1Hole2 = new G4PVPlacement(0, G4ThreeVector(0, 0, 90*CLHEP::mm),
                      lAlBox1Hole1, "AlBox1Hole2", lAlBox1, false, 0, checkOverlaps);


// --- iron block ---
sIronBlockOut = new G4Box("IronBlockOut",
                      220*CLHEP::mm/2, 220*CLHEP::mm/2, 25*CLHEP::mm/2);
sIronBlockIn  = new G4Tubs("IronBlockIn", 0, 38*CLHEP::mm, 25*CLHEP::mm/2,
                          startAngle, spanningAngle);

sIronBlock = new G4SubtractionSolid("IronBlock", sIronBlockOut, sIronBlockIn);
lIronBlock = new G4LogicalVolume(sIronBlock, iron, "IronBlock");
pIronBlock = new G4PVPlacement(0, G4ThreeVector(0, 0, -3537.5*CLHEP::mm),
                      lIronBlock, "IronBlock", lRoomIn, false, 0, checkOverlaps);



// 3rd section - 2nd Aluminium tube

// --- 2nd Aluminium tube ---
sAlTube2Out = new G4Tubs("AlTube2Out", 0, 38*CLHEP::mm, 450*CLHEP::mm/2,
                        startAngle, spanningAngle);
sAlTube2In = new G4Tubs("AlTube2In", 0, 36*CLHEP::mm, 450*CLHEP::mm/2,
                        startAngle,spanningAngle);

sAlTube2 = new G4SubtractionSolid("AlTube2", sAlTube2Out, sAlTube2In)
lAlTube2 = new G4LogicalVolume(sAlTube2, aluminium, "AlTube2");
pAlTube2 = new G4PVPlacement(0, G4ThreeVector(0, 0, -3325*CLHEP::mm),
                      lAlTube2, "AlTube2", lRoomIn, false, 0, checkOverlaps);



// 4th section - 2nd Aluminium box

// --- 2nd Aluminium box ---
sAlBox2Out = new G4Box("AlBox2Out", 200*CLHEP::mm/2, 200*CLHEP::mm/2, 592*CLHEP::mm/2);
sAlBox2In = new G4Box("AlBox2In",   160*CLHEP::mm/2, 160*CLHEP::mm/2, 552*CLHEP::mm/2);

sAlBox2 = new G4SubtractionSolid("AlBox2", sAlBox2Out, sAlBox2In);
lAlBox2 = new G4LogicalVolume(sAlBox2, aluminium, "AlBox2");
pAlBox2 = new G4PVPlacement(0, G4ThreeVector(0., 0., -2804*CLHEP::mm),
                                  lAlBox2, "AlBox2", lRoomIn, false, 0, false);
```

```cpp


    // - holes for beamline
    // copy from the first box
    pAlBox2Hole1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -286*CLHEP::mm),
                          lAlBox1Hole1, "AlBox2Hole1", lAlBox2, false, 0, checkOverlaps);

    sAlBox2Hole2 = new G4Tubs("AlBox2Hole2", 0, 20*CLHEP::mm, 20*CLHEP::mm/2,
                              startAngle, spanningAngle);
    lAlBox2Hole2 = new G4LogicalVolume(sAlBox2Hole2, airMaterial, "AlBox2Hole2");
    pAlBox2Hole2 = new G4PVPlacement(0, G4ThreeVector(0., 0., 286*CLHEP::mm),
                          lAlBox2Hole2, "AlBox2Hole2", lAlBox2, false, 0, checkOverlaps);


    // --- 2nd collimator ---
    sScatterCollim1Out = new G4Box("ScatterCollimOut",
                                      160*CLHEP::mm/2, 160*CLHEP::mm/2, 10*CLHEP::mm/2);
    sScatterCollim1In  = new G4Tubs("ScatterCollimIn", 0, 20*CLHEP::mm, 10*CLHEP::mm/2,
                                      startAngle, spanningAngle);

    sScatterCollim1 = new G4SubtractionSolid("ScatterCollim1",
                                          sScatterCollim1Out, sScatterCollim1In);
    lScatterCollim1 = new G4LogicalVolume(sScatterCollim1, brass, "ScatterCollim1");
    pScatterCollim1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -261*CLHEP::mm),
                      lScatterCollim1, "ScatterCollim1", lAlBox2, false, 0, checkOverlaps);


    // --- dose monitors ----

    // -> 1st dose monitor <-
    sDoseMonitor1 = new G4Box("DoseMonitor1",
                          160*CLHEP::mm/2, 160*CLHEP::mm/2, 37*CLHEP::mm/2);
    lDoseMonitor1 = new G4LogicalVolume(sDoseMonitor1, airMaterial, "DoseMonitor1");
    pDoseMonitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -225*CLHEP::mm),
                      lDoseMonitor1, "DoseMonitor1", lAlBox2, false, 0, checkOverlaps);

    // - PMMA plastic block - contains all the other elements
    sPlasticBlockOut = new G4Box("PlasticBlock1Out",
                              160*CLHEP::mm/2, 160*CLHEP::mm/2, 37*CLHEP::mm/2);
    sPlasticBlockIn  = new G4Tubs("PlasticBlock1In", 0, 30*CLHEP::mm, 37*CLHEP::mm/2,
                              startAngle, spanningAngle);

    sPlasticBlock1 = new G4SubtractionSolid("PlasticBlock1",
                                          sPlasticBlockOut, sPlasticBlockIn);
    lPlasticBlock1 = new G4LogicalVolume(sPlasticBlock1, PMMA, "PlasticBlock1");
    pPlasticBlock1 = new G4PVPlacement(0, G4ThreeVector(0, 0, 0),
                  lPlasticBlock1, "PlasticBlock1", lDoseMonitor1, false, 0, checkOverlaps);


    // -- 1st perspex layer - 8 mm
    sPerspex1MonitorBlock = new G4Box("Perspex1Monitor1Block",
                                          160*CLHEP::mm/2, 160*CLHEP::mm/2, 8*CLHEP::mm/2);
    sPerspex1MonitorHole  = new G4Tubs("Perspex1Monitor1Hole",
                                                0, 30.*CLHEP::mm, 8*CLHEP::mm/2,
                                                    startAngle, spanningAngle);

    sPerspex1Monitor1 = new G4SubtractionSolid("Perspex1Monitor1",
                                      sPerspex1MonitorBlock, sPerspex1MonitorHole);
    lPerspex1Monitor1 = new G4LogicalVolume(sPerspex1Monitor1, PMMA, "Perspex1Monitor1");
    pPerspex1Monitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -5.905*CLHEP::mm),
          lPerspex1Monitor1, "Perspex1Monitor1", lDoseMonitor1, false, 0, checkOverlaps);


    // HV foil 1
    // 0.005 mm Mylar lined with 0.001 mm Al

    // -- 1st mylar layer - 0.005 mm
    sMylar1Monitor1 = new G4Box("Mylar1Monitor1",
                              160*CLHEP::mm/2, 160*CLHEP::mm/2, 0.005*CLHEP::mm/2);
    lMylar1Monitor1 = new G4LogicalVolume(sMylar1Monitor1, mylar, "Mylar1Monitor1");
    pMylar1Monitor1 = new G4PVPlacement(0, G4ThreeVector(0 ,0, -1.9025*CLHEP::mm),
                              lMylar1Monitor1, "Mylar1Monitor1", lDoseMonitor1, false, 0);
```

```


// -- 1st aluminium layer - 0.001 mm
sAl1Monitor1 = new G4Box("Al1Monitor1",
                          160*CLHEP::mm/2, 160*CLHEP::mm/2, 0.001*CLHEP::mm/2);
lAl1Monitor1 = new G4LogicalVolume(sAl1Monitor1, aluminium, "Al1Monitor1");
pAl1Monitor1 = new G4PVPlacement(0, G4ThreeVector(0,0,-1.8005*CLHEP::mm),
                        lAl1Monitor1, "Al1Monitor1", lDoseMonitor1, false, 0);


// -- 2nd perspex layer - 1 mm
sPerspex2MonitorBlock = new G4Box("Perspex2Monitor1Block",
                              160*CLHEP::mm/2, 160*CLHEP::mm/2, 1*CLHEP::mm/2);
sPerspex2MonitorHole  = new G4Tubs("Perspex2Monitor1Hole",
                              0, 30*CLHEP::mm, 1*CLHEP::mm/2,
                                  startAngle, spanningAngle);

sPerspex2Monitor1 = new G4SubtractionSolid("Perspex2Monitor1",
                              sPerspex2MonitorBlock, sPerspex2MonitorHole);
lPerspex2Monitor1 = new G4LogicalVolume(sPerspex2Monitor1, PMMA, "Perspex2Monitor1");
pPerspex2Monitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, -1.3*CLHEP::mm),
        lPerspex2Monitor1, "Perspex2Monitor1", lDoseMonitor1, false, 0, checkOverlaps);


// -- guard ring - provides air gap
sGuardRing1 = new G4Tubs("GuardRing1", 30*CLHEP::mm, 50*CLHEP::mm, 1.6*CLHEP::mm/2,
                          startAngle, spanningAngle);
lGuardRing1 = new G4LogicalVolume(sGuardRing1, brass, "GuardRing1");
pGuardRing1 = new G4PVPlacement(0, G4ThreeVector(0, 0, 0), lGuardRing1,
                        "GuardRing1", lDoseMonitor1, false, 0, checkOverlaps);


// -- 3rd perspex layer - 1 mm
pPerspex3Monitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, 1.3*CLHEP::mm),
       lPerspex2Monitor1, "Perspex3Monitor1", lDoseMonitor1, false, 0, checkOverlaps);


// -- 2nd aluminium layer
pAl2Monitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, 1.8005*CLHEP::mm),
          lAl1Monitor1, "Al2Monitor1", lDoseMonitor1, false, 0, checkOverlaps);


// -- 2nd mylar layer
pMylar2Monitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, 1.9025*CLHEP::mm),
          lMylar1Monitor1, "Mylar2Monitor1", lDoseMonitor1, false, 0, checkOverlaps);


// -- 4th perspex layer
pPerspex4Monitor1 = new G4PVPlacement(0, G4ThreeVector(0, 0, 5.905*CLHEP::mm),
        lPerspex1Monitor1, "Perspex4Monitor1", lDoseMonitor1, false, 0, checkOverlaps);



// -> 2nd dose monitor <-
pDoseMonitor2 = new G4PVPlacement(0, G4ThreeVector(0, 0, -169.5*CLHEP::mm),
                lDoseMonitor1, "DoseMonitor2", lAlBox2, false, 0, checkOverlaps);


// --- Tungsten cross-wires ---

// - Fixture
sWiresFixtureOut = new G4Box("WiresFixtureOut",
                          160*CLHEP::mm/2, 160*CLHEP::mm/2, 40*CLHEP::mm/2);
sWiresFixtureIn  = new G4Box("WiresFixtureIn",
                          140*CLHEP::mm/2, 140*CLHEP::mm/2, 40*CLHEP::mm/2);

sWiresFixture = new G4SubtractionSolid("WiresFixture",
                              sWiresFixtureOut, sWiresFixtureIn);
lWiresFixture = new G4LogicalVolume(sWiresFixture, aluminium, "WiresFixture");
pWiresFixture = new G4PVPlacement(0, G4ThreeVector(0, 0, 140*CLHEP::mm),
                  lWiresFixture, "WiresFixture", lAlBox2, false, 0, checkOverlaps);

```

```
// Need to rotate wires as axis normally lies in z
// Rotate wire 1 around x axis
G4RotationMatrix xRot90deg = G4RotationMatrix();
xRot90deg.rotateX(M_PI/2*CLHEP::rad);
G4Transform3D rotWire1 = G4Transform3D(xRot90deg, G4ThreeVector(0, 0, 19.8*CLHEP::mm));

// Rotate wire 2 around y axis
G4RotationMatrix yRot90deg = G4RotationMatrix();
yRot90deg.rotateY(M_PI/2*CLHEP::rad);
G4Transform3D rotWire2 = G4Transform3D(yRot90deg, G4ThreeVector(0, 0, 19.8*CLHEP::mm));

// Define wires
sCrossWire1 = new G4Tubs("CrossWire1", 0, 0.4*CLHEP::mm/2, 140*CLHEP::mm/2,
                         startAngle, spanningAngle);
lCrossWire1 = new G4LogicalVolume(sCrossWire1, tungsten, "CrossWire1");
pCrossWire1 = new G4PVPlacement(rotWire1, lCrossWire1, "CrossWire1",
                         lWiresFixture, false, 0, false);

pCrossWire2 = new G4PVPlacement(rotWire2, lCrossWire1, "CrossWire2",
                               lWiresFixture, false, 0, false);


// --- Nozzle ---
sNozzle = new G4Tubs("Nozzle", 17*CLHEP::mm, 20*CLHEP::mm, 86.5*CLHEP::mm/2,
                      startAngle, spanningAngle);
lNozzle = new G4LogicalVolume(sNozzle, brass, "Nozzle");
pNozzle = new G4PVPlacement(0, G4ThreeVector(0, 0, 319.25*CLHEP::mm),
                      lNozzle, "Nozzle", lAlBox2, false, 0, checkOverlaps);


// --- Shielding ---
// x = 100 mm - 37/2 mm
sShielding1 = new G4Box("Shielding1",
                      37*CLHEP::mm/2, 230*CLHEP::mm/2, 1140*CLHEP::mm/2);
lShielding1 = new G4LogicalVolume(sShielding1, boratedPlastic, "Shielding1");
pShielding1 = new G4PVPlacement(0,
                  G4ThreeVector(128.5*CLHEP::mm, 15*CLHEP::mm, -3374*CLHEP::mm),
                  lShielding1, "Shielding1", lRoomIn, false, 0, checkOverlaps);

// x = - 100 mm + 37/2 mm
pShielding2 = new G4PVPlacement(0,
                  G4ThreeVector(-128.5*CLHEP::mm, 15*CLHEP::mm, -3374*CLHEP::mm),
                  lShielding1, "Shielding2", lRoomIn, false, 0, checkOverlaps);


// --- Detector ---
sDet = new G4Box("Detector", detSizeXY/2, detSizeXY/2, detSizeZ/2);
lDet = new G4LogicalVolume(sDet, fWorldMaterial, "Detector");
pDet = new G4PVPlacement(0, detPosition, lDet, "Detector",
                                  lRoomIn, false, 0, checkOverlaps);



// Absorber -- fAbsorSizeXY etc. set in proton.mac
sAbsor = new G4Box("Absorber", fAbsorSizeXY/2, fAbsorSizeXY/2, fAbsorSizeZ/2);
lAbsor = new G4LogicalVolume(sAbsor, fAbsorMaterial, "Absorber");
pAbsor = new G4PVPlacement(0, G4ThreeVector(), lAbsor, "Absorber",
                                          lDet, false, 0, checkOverlaps);

// Layer
if (fLayerNumber > 0) {
  fLayerSizeZ = fAbsorSizeZ/fLayerNumber;
  G4Box* sLayer = new G4Box("Layer", fLayerSizeXY/2, fLayerSizeXY/2, fLayerSizeZ/2);
  lLayer = new G4LogicalVolume(sLayer, fAbsorMaterial, "Layer");
  pLayer = new G4PVReplica("Layer", lLayer, lAbsor, kZAxis, fLayerNumber, fLayerSizeZ);

  fLayerMass = fLayerSizeXY*fLayerSizeXY*fLayerSizeZ*(fAbsorMaterial->GetDensity());
}

PrintParameters();


```

6.4.2 SteppingAction.cc

```cpp
// M Hentz, 2016

#include "SteppingAction.hh"
#include "SteppingActionMessenger.hh"
#include "G4Step.hh"
#include "G4StepPoint.hh"
#include "G4VisExtent.hh"
#include "DetectorConstruction.hh"
#include "RunAction.hh"
#include "Randomize.hh"
#include "G4EventManager.hh" // for eventID

SteppingAction::SteppingAction(DetectorConstruction* det, RunAction* RuAct)
:G4UserSteppingAction(), fDetector(det), fRunAction(RuAct), steppingActionMessenger(0)
{
  steppingActionMessenger = new SteppingActionMessenger(this);
}

SteppingAction::~SteppingAction()
{ }

void SteppingAction::UserSteppingAction(const G4Step* step)
{
  G4double edep = step->GetTotalEnergyDeposit();
  if (edep <= 0.) return;

  // record energy deposited in the step
  fRunAction->FillEdep(edep);

  // increment counter for primary particle steps
  if (step->GetTrack()->GetTrackID() == 1)
  {
    fRunAction->AddPrimaryStep();
  }

  // access the end points of the step
  G4StepPoint* prePoint  = step->GetPreStepPoint();
  G4StepPoint* postPoint = step->GetPostStepPoint();

  // get the particle name
  G4String particleName = step->GetTrack()->GetDefinition()->GetParticleName();

  // get the parentID: if not 0 then particle is a secondary
  G4int parentID = step->GetTrack()->GetParentID();

  // check if particle is a proton
  //  -> see 'ternary operators' if not familiar with this
  bool isProton = (particleName == "proton") ? true : false;

  // for debugging - prints bools as 'true' or 'false'
  G4cout << std::boolalpha;

  // check if protons are inside tracking area
  bool isInside = ((abs(xposPost) <= 100 || abs(yposPost) <= 100))
                  ? true : false;

  if (isTracked && isProton && isInside)
  {
    // get kinetic energy to write to file
    G4double kePost  = postPoint->GetKineticEnergy();

    // get position in x, y
    G4double xposPost = postPoint->GetPosition().x();
    G4double yposPost = postPoint->GetPosition().y();

    // since the source is at -4200 mm in z
    // need to shift by 4200 to get distance from source
    // -- relative to centre of room
    G4double zposPreCentre = prePoint->GetPosition().z();
    G4double zShift  = 4200.;
    G4double zposPre = zposPreCentre + zShift;

    G4double zposPostCentre = postPoint->GetPosition().z();
    G4double zposPost = zposPostCentre + zShift;

    // get direction of travel
    G4ThreeVector direction = prePoint->GetMomentumDirection();

    // angle between x-projection and z-axis for emittance
    // factor of 1000 to convert to mrad
    // the approximation angle = x/z would also work
    G4double angle = atan(direction.x()/direction.z())*1000;

    // iterate through all the desired boundaries
    for (int z_boundary : trackingDistVec)
    {

      // score when particle traverses boundary
      // make sure very first particle is recorded
      if (
          (zposPre < z_boundary && zposPost >= z_boundary) ||
          (zposPre == 0 && zposPre == z_boundary)
      )
      {
        // set filenames and open output files
        std::string output_filename = "data/output_z"
                                    + std::to_string(z_boundary) + ".txt";
        std::ofstream outputfile(output_filename, std::ios::app);

        if(outputfile.is_open())
        {
          // only score forward moving particles
          if (direction.z() > 0)
          {
            // write to file
            // the particles' current position is in postPoint
            outputfile << parentID << " "
                       << xposPost << " " << yposPost << " " << zposPost << " "
                       << angle    << " "
                       << kePost << G4endl;
          }
          outputfile.close();

        } // if is_open

      } // boundary check

    } // loop


    // fill layers of the absorber
    G4String prePointVol = prePoint->GetTouchableHandle()->GetVolume()->GetName();

    if (prePointVol == "Layer")
    {
      G4int layerNb = prePoint->GetTouchableHandle()->GetCopyNumber();
      fRunAction->FillLayerEdep(layerNb, edep);
    }

  } // tracking

} // UserSteppingAction
```

6.4.3 SteppingAction.hh

```
// M Hentz, 2016

#ifndef SteppingAction_h
#define SteppingAction_h 1

#include "G4UserSteppingAction.hh"
#include "G4VProcess.hh"

class DetectorConstruction;
class RunAction;
class SteppingActionMessenger;

class SteppingAction : public G4UserSteppingAction
{
public:
  SteppingAction(DetectorConstruction*, RunAction*);
  virtual ~SteppingAction();

  virtual void UserSteppingAction(const G4Step*);

  inline void InitialiseFiles();

  void SetTracking(G4String trackingString)
  {
    // check if tracking is switched on and act accordingly
    if (trackingString == "All")
    {
      isTracked = true;
      trackSingle = false;

      // this vector contains all the predefined tracking positions
      // most correspond to a position just before or after a component
      // - edit at will
      std::vector<int> trackingDistFiller(14);
      trackingDistFiller = { 0, 45, 79, 81, 299, 307, 400,
                             800, 1099, 1671, 1759, 1800, 1839, 1881 };

      // trackingDistVec is available to all of SteppingAction.cc
      trackingDistVec = trackingDistFiller;
    }
    else if (trackingString == "None")
    { isTracked = false; }
    else
    {
      isTracked = true;
      trackSingle = true;

      // convert string to int
      zposTracker = std::stoi(trackingString);

      // set to position given in macro
      std::vector<int> trackingDistFiller(1);
      trackingDistFiller = { zposTracker };

      trackingDistVec = trackingDistFiller;
    }

    InitialiseFiles();
  }




private:
  DetectorConstruction* fDetector;
  RunAction*            fRunAction;

  G4int     nentries;

  bool first = true;
  bool isTracked;
  bool trackSingle;

  std::vector<int> scoringDistVec;
  int zposTracker;

  std::map<std::string,double> componentMap;

  SteppingActionMessenger*  steppingActionMessenger;

};

// initialises all the files that will be written to before the run
inline void SteppingAction::InitialiseFiles()
{
  std::string trackerPositions = "trackerPositions.txt";
  std::ofstream trackerfile(trackerPositions, std::ios::app);

  bool write_header = true;

  for (int distance : trackingDistVec)
  {

    // write all tracker positions to a file
    if(trackerfile.is_open())
    {
      if (write_header)
      {
        trackerfile << "# tracker positions in z [mm]" << G4endl;
        write_header = false;
      }

      trackerfile << distance << G4endl;
    }

    std::string output_filename = "data/output_z" + std::to_string(distance) + ".txt";
    std::ofstream outputfile(output_filename, std::ios::app);

    if(outputfile.is_open())
    {
      outputfile << "# zposition: " << distance << G4endl;
      outputfile << "# parentID, x, y, z [mm], theta [mrad], ke [MeV]" << G4endl;

      outputfile.close();
    }
  }

  trackerfile.close();
}


#endif

```